uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería en Tecnologías Industriales

Trabajo Fin de Grado

Aplicaciones marinas de Fast Marching

Autor:

Carlos Avendaño Palacios

Tutor:

Luis Santiago Garrido Bullón

Curso 2018-2019

Resumen

Desde que el ser humano fue capaz de viajar, una de las preocupaciones más importantes ha sido cómo hallar la mejor trayectoria posible entre el origen y el destino. En los últimos años, gracias al desarrollo de la computación y a la optimización de algoritmos, se han desarrollado potentes métodos capaces de hallar la ruta más segura y rápida posible. Entre estos, uno de los más célebres es el Fast Marching (FM), el cual será utilizado a lo largo de todo el proyecto para explorar sus ventajas y limitaciones, partiendo desde los casos más básicos y teóricos hasta los más complejos y reales.

Este proceso puede ser empleado para cualquier tipo de recorrido, ya sea por aire, mar, tierra, etc. En el presente trabajo nos centraremos en travesías marítimas, buscando como objetivo final trazar trayectorias en mares y océanos reales con las corrientes que existen en tales zonas.

Palabras clave: Fast Marching, Fast Marching Square, FM, FM^2.

Agradecimientos

En primer lugar, quiero dar las gracias a mi tutor Santiago Garrido por darme esta oportunidad y haberme ayudado en infinidad de ocasiones, sin su apoyo hubiese sido imposible.

Agradecer a mis amigos Miguel Espinilla y Julio Revuelta su ayuda en momentos críticos del desarrollo del proyecto.

Y agradecer a la educación pública en general, y a la universidad UC3M en particular, por la educación de calidad que he tenido estos cuatro años de carrera y las experiencias inolvidables que he vivido. Y, desde luego, a todos los grandes profesores que me han traído hasta aquí.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

Capítulo 3:

Capítulo 4 – Primera fase:

Capítulo 4 – Segunda fase:

Capítulo 4 – Tercera fase:

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Objetivo

El objetivo del proyecto es analizar los resultados del algoritmo del Fast Marching (a partir de ahora se llamará Fast Marching o FM indistintamente) aplicado como método de resolución en la planificación de trayectorias [1]. El objetivo final será utilizar el FM en mapas con corrientes reales de modo que el trabajo aquí desarrollado pudiera validar un posible uso del método para la navegación habitual de barcos.

Haciendo uso de Matlab como herramienta básica en todo el proceso, y ayudándonos de fuentes externas de datos -que serán correctamente citadas-, analizamos el comportamiento de barcos en rutas marinas según las corrientes y la geografía.

1.2 Estructura

En el capítulo 2 me centraré en introducir el estado del arte hoy en día, es decir, qué métodos principales se utilizan a la hora de planificar rutas y las principales diferencias entre ellos, teniendo como objetivo poner en un contexto general al FM.

En el capítulo 3 realizaré una breve introducción histórica de los más importantes hitos científicos que han hecho posible el desarrollo del FM y mostraré un ejemplo muy básico, ya que considero que es lo más intuitivo y eficaz para introducir el proceso. Después analizaré el funcionamiento del algoritmo y del método en su dimensión teórica, tratando siempre de explicarlo de la forma más sencilla posible en vez de en términos puramente matemáticos, ya que explicar el complejo proceso matemático que hay detrás [2], aunque podría ser interesante, no es el objeto de este trabajo ni es necesario para el correcto desarrollo del mismo al ser una aplicación experimental.

El capítulo 4, el de mayor extensión e importancia, es donde muestro los resultados de todos estos meses de experimentación y donde se podrá ver el funcionamiento del método y sus ventajas e inconvenientes, explicando qué limitaciones y problemas me encontré, a la vez que las soluciones que tuve que desarrollar. Todo ello lo iré documentando con imágenes reales de lo que fui obteniendo a lo largo del proceso. Comenzaré la primera fase con ejemplos muy simples del FM y continuaré en la segunda explicando el funcionamiento de las corrientes, hasta acabar en la tercera fase con mapas de mareas reales habiendo obteniendo previamente los datos de corriente en todas sus coordenadas.

En los capítulos 5, 6 y 7 hablo, respectivamente, de futuras ampliaciones de este trabajo, del presupuesto del proyecto y, por último, realizo una conclusión en la cual reflexiono sobre todo lo que he aprendido a lo largo de este curso trabajando en el proyecto, tanto a nivel académico como personal.

2. Estado del arte

2.1. Planificación de trayectorias

El desarrollo de algoritmos que buscan una trayectoria óptima sigue siendo un campo de investigación abierto y, en general, un método puede ser mejor que otro dependiendo de la situación en la que nos encontremos y de lo que busquemos, ya que la trayectoria deseada tiene que presentar unas características de seguridad, suavidad y tiempo de llegada; y la relación entre estas características puede variar según el método: generalmente, si queremos reducir al mínimo el tiempo de llegada, esto se hará a costa de perder suavidad y distancia de seguridad, y así con el resto de características.

Hay un consenso generalizado respecto al hecho de que los algoritmos que tienen una alta carga computacional dan mejores resultados mientras que los que tienen una carga computacional menor dan como resultado trayectorias más bruscas, lo que es intuitivo ya que al hacer un mayor número de operaciones (e iteraciones) se consiguen resultados más precisos.

Basándonos en criterios geométricos, podríamos dividir los métodos en los siguientes grupos: [3]

- Métodos combinatorios: obtienen las trayectorias tras obtener toda la información necesaria para ello, aplicando la optimización combinatoria; uno de sus ejemplos más importantes son las hojas de ruta, que acaban convirtiendo el proceso en la resolución de un grafo de una manera similar al algoritmo de Dijkstra.
- Métodos basados en muestreo: son métodos incrementales en los cuales se va generando desde el origen (incluso de forma aleatoria en algunas variantes) posibles trayectorias hasta alcanzar el objetivo. En general, no encuentran el óptimo sino que se quedan con una solución heurística. Uno de los métodos más célebres de este tipo es Árbol de Exploración rápida aleatoria (RTT por sus siglas en inglés) [4].
- Métodos basados en espacios discretos: una variante del método de muestreo con un coste computacional mayor, en este caso primero se considera al espacio como discreto a través de su división en rejillas o celdas; en principio al hacerlo discreto se pierde precisión (sucede de similar manera al pasar de analógico a digital), pero todo depende del tamaño de las celdas y de lo que queramos obtener (sería el equivalente al número de bits). De este modo, se va estudiando la relación entre cada celda y sus vecinas asemejándolo a un campo potencial, donde hay unos trayectos más favorables que otros. El Fast Marching, que será explicado con más profundidad en el siguiente capítulo, se encuentra dentro de esta categoría, asignando a cada celda (inicialmente rectangular) un coste o tiempo de llegada, y de este modo se van generando unos caminos más rápidos que otros.

Hoy en día se sigue investigando para optimizar los métodos existentes, en algunos casos mezclándolos entre ellos. Uno de los puntos críticos en la actualidad es la detección de colisiones, ya que generalmente se tienen que emplear algoritmos externos para poder detectarlas.

2.2 Diseño de soluciones.

Para trabajar con este método empleamos varias *toolbox* de Matlab: principalmente la *toolbox* oficial de este método [5] con variaciones introducidas tanto por mi tutor como por mí y otros compañeros; además de otras *toolbox* de FM aplicado a corrientes u otras que sirvieron de apoyo como *flypath* [6] o *nctoolbox* [7].

3. Fast Marching

Fast Marching es un potente método matemático, desarrollado en 1996 por Sethian (1954, California), que será utilizado en este proyecto para poder simular las rutas más rápidas y seguras entre el origen y el destino; en el presente caso, lo aplicaremos a trayectos sobre la superficie marina con corrientes, pero también sería aplicable al vuelo de un avión, al movimiento de grupos de robots [8], a la simulación de la propagación de un incendio, etc.

3.1. Introducción histórica

El estudio de la propagación de ondas ha sido un tema de interés y debate para científicos e ingenieros desde hace varios siglos; uno de los investigadores más célebres en este campo fue es escocés James Clerk Maxwell (1831-1879), quien formuló sus famosas cuatro ecuaciones (aumentaría el número si consideramos sus respectivas formulaciones escalares) capaces de condensar todo el conocimiento previo de otros grandes pensadores como Faraday, Gauss o Coulomb.

Este conjunto de ecuaciones es básico en gran parte de las diversas ingenierías y ciencias debido a que son capaces de describir de forma elegante (esto es, de forma concisa y precisa) la práctica totalidad de los fenómenos electromagnéticos [9] [10]. Brevemente, estas cuatro ecuaciones son:

$$\oint_S \vec{E} \cdot d\vec{S} = \frac{q}{\varepsilon_0}$$

Ley de Gauss para la electricidad

La cual establece que el flujo de campo eléctrico que fluye a través de una superficie cerrada es proporcional a la carga neta encerrada en esa superficie dividida por la permitividad del vacío.

$$\oint_S \vec{B} \cdot d\vec{S} = 0$$

Ley de Gauss para el magnetismo

Esta segunda ecuación muestra que el flujo de campo magnético a través de cualquier superficie cerrada es cero, confirmando el hecho de que no existen polos magnéticos aislados.

$$\oint \vec{\mathrm{E}} \cdot \mathrm{d}\vec{l} = -\int_s \frac{d\vec{B}}{\mathrm{dt}} \cdot d\vec{\mathrm{S}}$$

Ley de Faraday-Lenz

Esta ecuación relaciona el campo magnético y el eléctrico a través de uno de los fenómenos más importantes en esta disciplina: la inducción electromagnética. En esencia, lo que implica es que la variación de un campo magnético genera un campo eléctrico; de hecho, esta propiedad es frecuentemente utilizada para generar electricidad, como en un generador.

$$\oint_c \vec{B} \cdot d\vec{l} = \mu_0 \int_s \vec{J} \cdot d\vec{S} + \mu_0 \varepsilon_0 \frac{d}{dt} \int_s \vec{E} \cdot d\vec{S}$$

Ley de Ampére

La última ecuación de Maxwell es la más compleja y ha tenido modificaciones históricas debido a que en su inicio no respetaba el principio de conservación de la carga y hubo que añadir nuevos términos; en esencia, muestra que un campo eléctrico variable genera un campo magnético [11].

Estas cuatro ecuaciones mostraron que el campo magnético y el eléctrico están siempre en estrecha relación; sin embargo, el legado del trabajo de Maxwell fue más allá del electromagnetismo llegando a muchas otras áreas y disciplinas como la óptica.

De estas ecuaciones surge y se deriva la ecuación de Eikonal, la cual trata de relacionar los rayos de luz con las ondas, es decir, la parte geométrica y la parte física de la óptica:

$$\nabla u| = F(x), \quad x \in \Omega$$

Ecuación de Eikonal

Dicha fórmula trata de representar la propagación de la luz en un medio no homogéneo; sin embargo, se encontraba con ciertos límites que son resueltos por el método del Fast Marching.

¿Qué límites eran estos? En nuestro caso, históricamente ha sido complicado compaginar los campos atractivos (como el electromagnético) con campos repulsivos pese a haber tratado de resolverlo a través de complejos desarrollos matemáticos. En el método del Fast Marching se propone juntar ambos tipos de campo del mismo modo a como se relacionan en la naturaleza, siguiendo el ejemplo de la luz.

Ahora bien, ¿por qué nos interesa estudiar el comportamiento de la luz para trazar viajes en barco?

La respuesta es intuitiva ya que la luz siempre sigue el camino más breve posible, según sabemos por el principio de Fermat, cuya formulación moderna podría ser:

"Un rayo de luz, viajando entre dos puntos, debe atravesar el camino cuya longitud *(tiempo)* sea estacionaria respecto a variaciones de dicho trayecto"
Traducción libre del principio de Fermat [12].

Es decir, si tienes una habitación cerrada a oscuras y abres una ventana, el camino que sigue la luz será el más rápido de entre todos los posibles; ocurriría lo mismo si alumbras con una linterna un lugar lejano, o si realizas un laberinto de luz con espejos; de hecho este comportamiento de la luz es frecuentemente utilizado en los ejemplos que tratan de explicar la teoría de la relatividad de una forma más visual e intuitiva.

Este hecho empírico está relacionado con la ley de la reflexión y de la refracción (leyes que también se derivan del principio de Fermat), además de otras propiedades de la luz, como el hecho de que nada puede ir más rápido que esta, o que su velocidad siempre es constante; propiedades que conocemos desde el siglo pasado gracias al trabajo de Albert Einstein (1879-1955), quien entre su inmenso legado nos dejó la fórmula más célebre de la historia:

$$E = mc^2$$

Expresión de la teoría de la relatividad

Dicha fórmula muestra que la relación entre masa y energía es, precisamente, la velocidad de la luz al cuadrado.

Volviendo al Fast marching, tratamos de imitar el comportamiento natural de la luz para obtener el camino más corto posible entre un punto inicial y otro final en vez de complicarnos en largos desarrollos matemáticos, sino más bien de una forma similar al algoritmo de Dikjstra.

Dicho algoritmo, desarrollado por el científico holandés Edsger Wybe Dijkstra (1930-2002), sirve para encontrar el camino más corto posible en una red de nodos o grafo con distintos valores (tiempo o coste de llegada) en cada arista; la diferencia radica en que nosotros trataremos de buscar el camino más breve posible en la propagación de una onda, que es más parecido a un campo continuo que a un campo discreto (grafo) formado por vértices (nodos) y aristas (caminos).

Una vez realizada esta breve introducción histórica, considero que lo más sencillo es mostrar primero el funcionamiento básico del FM con un ejemplo antes de continuar con el desarrollo teórico, pues podría parecer más complejo de lo que realmente es a la hora de aplicarlo.

3.2. Ejemplo *Room*

El primer paso en toda aplicación del FM siempre será tener un mapa en blanco y negro (llamado W_0) que representa el espacio donde planificaremos el trayecto. Ahora bien, una vez lo tenemos seleccionado, ¿qué es lo que vamos obteniendo al utilizar el método del Fast Marching? La mejor manera de explicarlo es con un caso real, pero antes hay que realizar ciertos comentarios:

Primero vamos a obtener un potencial repulsivo, W(x), también conocido como matriz de dificultad o lentitud, que será usado a modo de índice de refracción [13] y mostrará la propagación de ondas y los lugares donde se encuentra con obstáculos. En los ejemplos, esto se traduce en que el color se va oscureciendo según se acerca a las paredes/obstáculos, y muestra la máxima velocidad en cada punto; por ello, la velocidad en las paredes será cero (representado por el color negro), e irá aumentando según se aleje de dichas barreras. Es decir, analiza a cuánta distancia está el pixel de obstáculo más cercano y penaliza aquellos puntos con obstáculos cerca.

Seguidamente, utilizando el potencial repulsivo como índice de refracción (entendido como la inversa de la velocidad), obtenemos el segundo potencial (o potencial atractivo), que muestra el tiempo de llegada a cada punto desde el origen. [14]

Vamos a mostrar esto con el ejemplo más habitual de funcionamiento: elegir un punto inicial y final. Dichos puntos se introducen en el código de Matlab, al igual que otras variables como el mapa, las corrientes, etc. Dicho código se mostrará en el apéndice, ya que ahora a modo de ejemplo vamos a mostrar solo los resultados más importantes.

Partimos de la imagen *Room*, una de las más básicas y útiles para comprobar el funcionamiento, y que será utilizada en más de una ocasión en este trabajo:

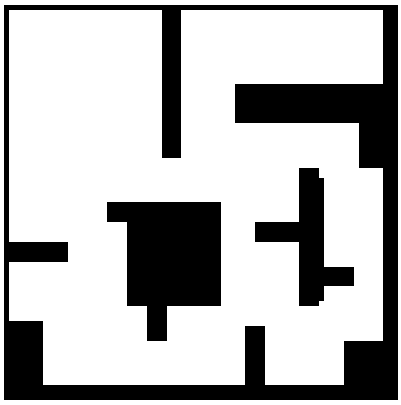


Figura 3.1: Imagen de Room

De esta imagen en blanco y negro, el FM obtiene el primer potencial, que es básicamente la traducción de dicho mapa en blanco y negro, de forma que tan solo los lugares en blanco se presenten como posibles (amarillo en la figura), mientras el resto tiene altura cero:

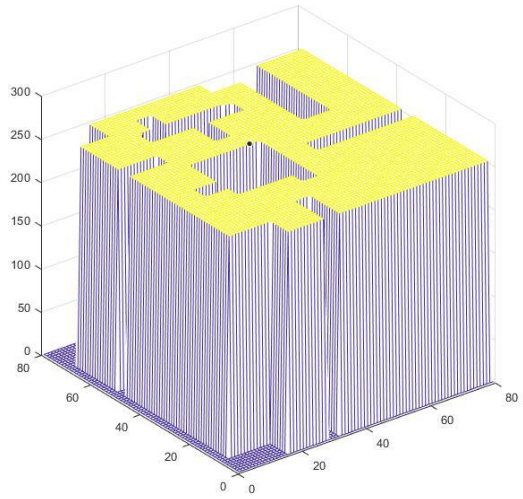


Figura 3.2: Primer potencial en Room

De hecho, la vista en planta de esta figura es, precisamente, la figura original de *Room* (con los colores cambiados):

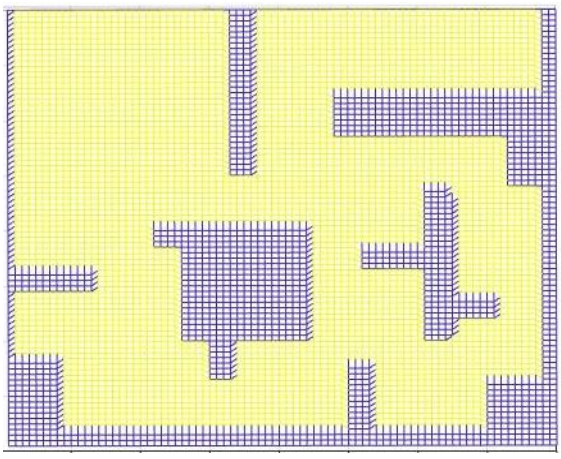


Figura 3.3: Planta del primer potencial en Room

Sin embargo, por seguridad, no se utiliza dicha figura sino que se emplea un método mejorado, el Fast Marching Square o FM^2,que no se basa solo en blanco y negro sino que incluye una escala de grises, evitando así acercarse demasiado a los obstáculos; en este caso el perfil sería el siguiente:

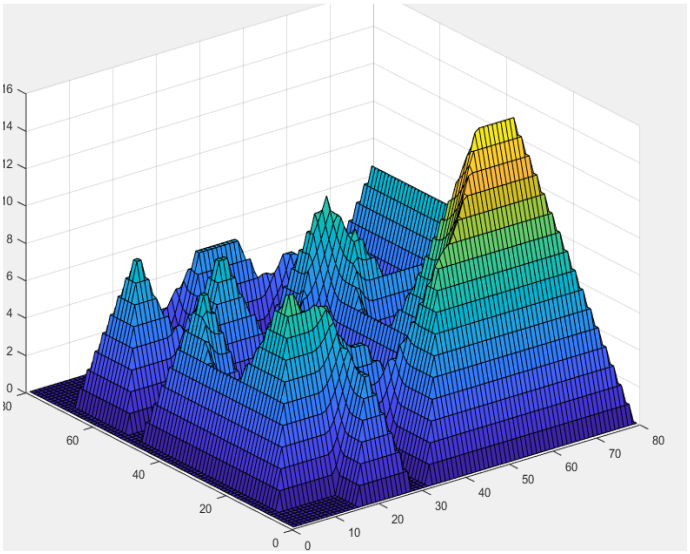


Figura 3.4: Primer potencial con FM^2 aplicado a Room

Es decir, en este caso el mapa ya no salta de puntos blancos a puntos negros de forma brusca, sino de forma continua y progresiva. Esto quizá se vea mejor de nuevo en la vista de planta de la figura anterior:

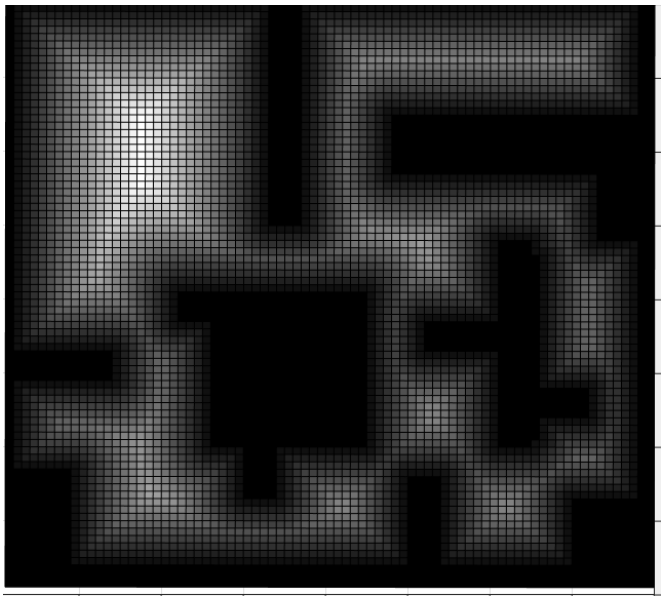


Figura 3.5: Planta del primer potencial de FM^2 aplicado a Room

Ahora la transición es más suave, por lo que también lo será la trayectoria finalmente escogida; esto se verá con más detenimiento cuando explique las diferencias y mejoras del FM^2 frente al FM tradicional.

El resultado que obtenemos es el siguiente:

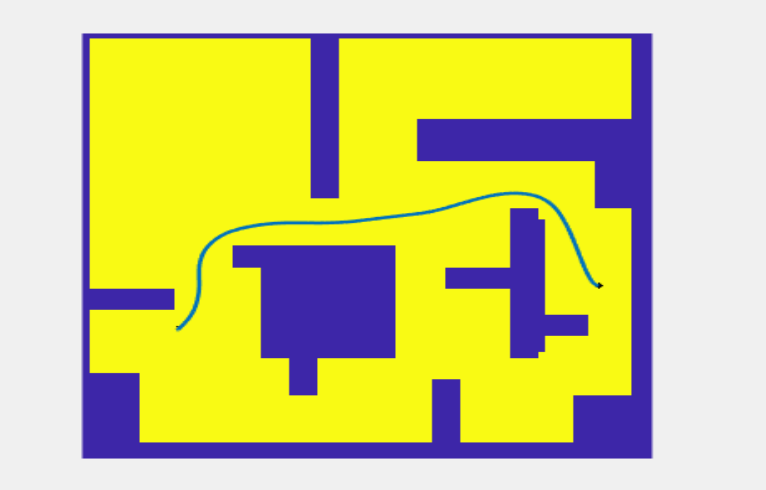


Figura 3.6: Trayectoria en Room

Evidentemente, los puntos inicial (izquierda) y final (derecha) fueron elegidos en el código con anterioridad. Ahora bien, ¿en base a qué elige esta trayectoria como la mejor, y no otra, como por ejemplo, por debajo de los obstáculos en vez de por encima? La respuesta está en el tiempo de llegada o segundo potencial:

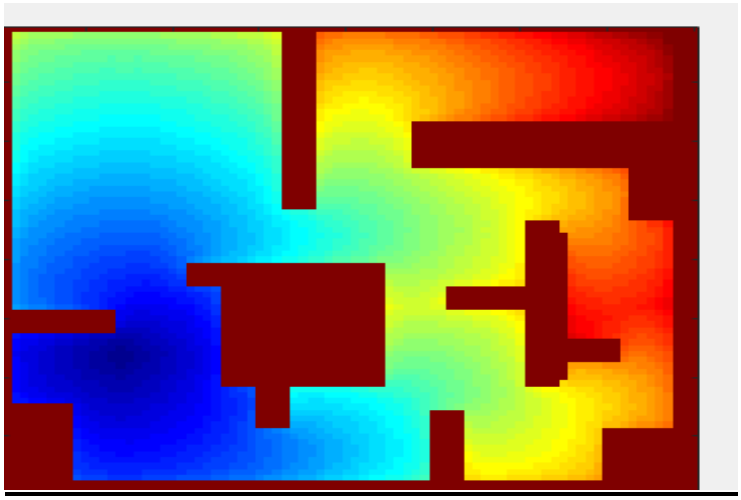


Figura 3.7: Tiempo de llegada de Room

El mapa muestra la propagación de la onda y cuánto tarda en llegar a cada punto desde el lugar inicial. La zona azul es la zona más cercana, y la roja la más lejana. El mejor camino no es ir por debajo de los obstáculos ya que tarda más en llegar, como se puede ver observando que el color rojo (representando mayor tiempo de llegada) aparece ligeramente antes. Esto se debe, entre otras cosas, a las corrientes que he seleccionado ya que de otra forma saldría como mejor opción el camino inferior, pero eso se verá más adelante.

Es interesante remarcar que, en la esquina superior derecha, la onda no llega realmente hasta el final de la pared sino que aparece difuminado según llega al extremo de esta. Esto se debe a una característica intrínseca del FM: este algoritmo se detiene cuando la onda alcanza el punto de destino y deja de expandirse; la explicación se encuentra en el hecho de que la velocidad de expansión es constante, por lo tanto una vez que alcance el punto final ya ha hallado el camino más breve posible, pues cualquier otro trayecto llegaría más tarde al destino.

Otra forma de ver esto, en vez de con los colores, es colocando el tiempo de llegada como tercer eje:

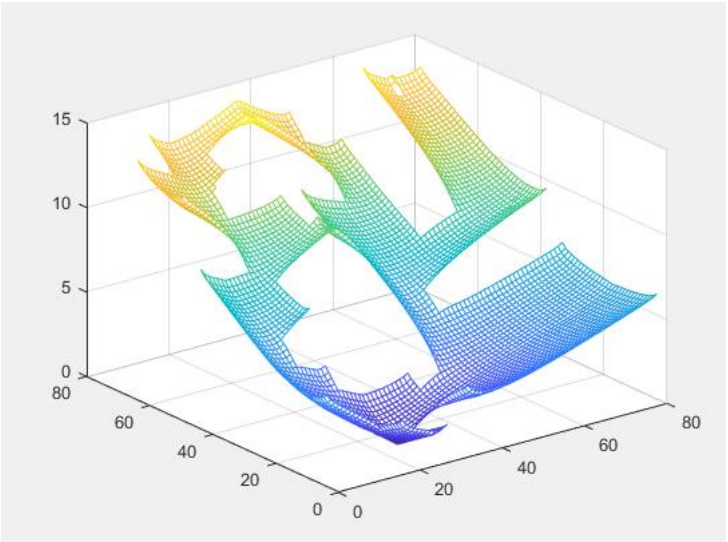


Figura 3.8: Tiempo de llegada vertical en Room

De este modo, el tiempo de llegada menor (color azul), corresponde a la zona más cercana al punto de inicio. Una forma de comprobarlo es volver a representar la vista de alzado de la figura, obteniendo la misma imagen de Room y su correspondiente propagación de onda, solo que en este caso con una paleta de colores distinta:

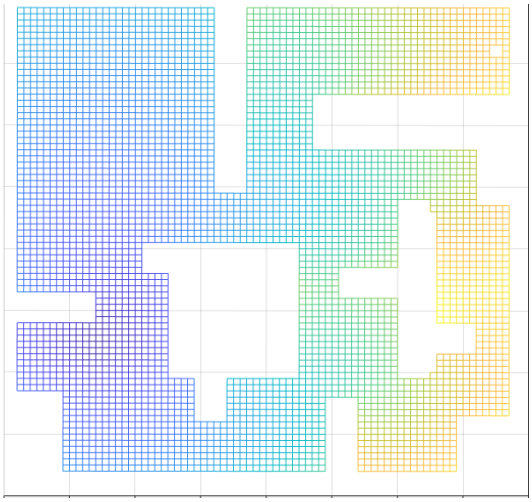


Figura 3.9: Planta de tiempo de llegada vertical en Room

3.3. Desarrollo teórico

Ahora bien, ¿cómo funciona el algoritmo de Fast Marching? Como se dijo en la introducción, no entraremos a explicar el complejo desarrollo matemático ya que podría llevar otro TFG entero y además no es necesario para las aplicaciones que haremos, pero sí es importante y útil tener una visión general:

Supongamos que estamos simulando la propagación del sonido del ladrido de un perro: la onda, desde el origen, va a comenzar a propagarse de forma simétrica y concéntrica como muestra la siguiente secuencia de imágenes.

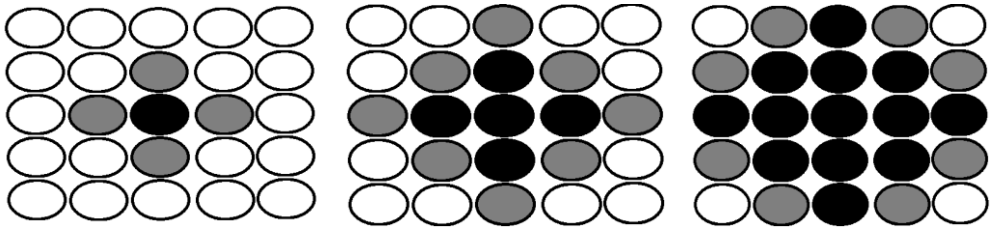


Figura 3.10: Propagación simplificada de una onda.

Los puntos negros representan lugares que han sido alcanzados por la onda (lugares en los que se ha escuchado el ladrido), los grises indican los lugares que serán o están alcanzados en la próxima iteración y, por último, los puntos blancos son aquellos a los que aún no ha llegado la propagación del sonido.

Es decir, el FM cataloga los puntos en tres categorías [15]:

- Puntos conocidos (llamados en artículos de divulgación *alive points* o *frozen points*): aquellos en los que sabemos la velocidad (el tiempo que ha tardado en llegar la onda desde su origen) y la situación (si es un obstáculo, se marcaría como tal). En este ejemplo son los puntos negros.
- Puntos alcanzados en la siguiente iteración (o *trial points*): son aquellos representados en gris. Al conjunto de trial points se le denomina *narrow band*. Según la fuente, en algunos casos estos puntos no son los que serán estudiados en la siguiente iteración sino los que están siendo estudiados en la actual; realmente, dado la duración infinitesimal de cada iteración, esta diferencia no es en absoluto trascendental.
- Puntos no conocidos (*unknown* o *far away points*). En blanco en la figura, son aquellos no alcanzados aún por la onda y que por tanto se desconoce el tiempo o coste de llegada hasta ellos. Si no se llega al destino antes, se convertirán en puntos conocidos en las futuras iteraciones.

La existencia de puntos sin información no es necesariamente negativa ya que, como vimos en el ejemplo, el algoritmo se detiene cuando hemos alcanzado nuestro destino, por lo que el resto de puntos a los que la onda aún no ha llegado no interesan ya que implicarían un camino más lento.

Si representamos el ejemplo anterior del ladrido del perro tomando el tiempo como eje vertical, obtenemos la siguiente figura (conocida como superficie de Lyapunov, caracterizada por su suavidad [16]) como potencial del tiempo de llegada, alejándose simétricamente respecto del punto inicial desde el cual nace la onda:

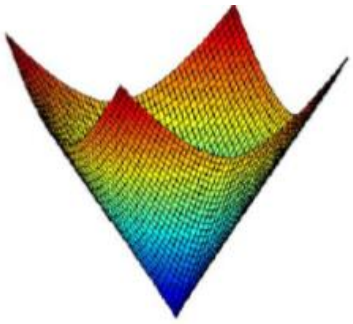


Figura 3.11: Tiempo de llegada con un solo foco y superficie sin obstáculos

Por este comportamiento binario es por lo que Fast Marching funciona teniendo como base mapas en blanco y negro, que representen obstáculos (negro) o espacio libre (blanco); en nuestro caso, el mar será la superficie blanca, mientras que la superficie negra representará la tierra, islas, otros barcos, etc.

Con este ejemplo, básico y recurrente, se entiende el FM como un proceso iterativo que va continuamente estudiando la propagación de una onda, resolviendo la ecuación de Eikonal en los puntos inmediatamente próximos hasta que llega al objetivo final, y gracias a las propiedades del potencial y a su pendiente, se obtiene el camino más breve posible.

Es decir, calcula el tiempo que la onda tarda en alcanzar cada punto del espacio estudiado (Eikonal), y de aquí deriva la velocidad en cada zona o, dicho de un modo más técnico y formal, calcula el gradiente de llegada T(x) que es precisamente la inversa de la velocidad F(x):

$$\frac{1}{F(x)} = |\nabla \mathrm{T}(\mathrm{x})|$$

Ecuación del Fast Marching [11]

Es importante remarcar que Fast Marching no funciona con velocidades negativas ya que la onda siempre está expandiéndose, es decir, esta no vuelve hacia atrás; y que, en su desarrollo, el algoritmo no tiene mínimo local, solo un mínimo global localizado, precisamente, en el punto de llegada.

Esta última característica del FM le convierte en un método con gran valor ya que otros algoritmos que tratan de encontrar la trayectoria óptima, al considerar mínimos locales en el proceso, provocan que el barco se quede atascado sin terminar el trayecto al encontrarse con el mínimo hallado y comprobar que en las celdas contiguas se empeora el resultado; en el FM se evita esto al solo tener un mínimo global en el punto de destino, asegurándose que el trayecto no termine hasta llegar al objetivo.

Sin embargo, el método tradicional de Fast Marching no era del todo seguro ya que, aunque daba la mejor solución posible en cuanto a la mínima distancia euclídea, las trayectorias pasaban excesivamente cerca de los obstáculos, lo que no era sensato ya que en la práctica implicaría que los barcos/robots/aviones pasarían prácticamente pegados al resto de cuerpos.

Para evitar el problema previamente comentado se desarrolló el Fast Marching Square (FM^2) [17] que soluciona este problema haciendo que la expansión de la onda esté influida según la distancia al obstáculo más cercano, penalizando los puntos más cercanos a favor de los más alejados, permitiendo de este modo incluir niveles de saturación que funcionan a modo de factor de seguridad haciendo que el camino se acerque más o menos a los límites físicos.

Otra de las ventajas del FM^2 es que obtiene un *feedback* constante ya que recalcula continuamente el trayecto de forma iterativa, lo que aumenta la seguridad ya que, si apareciera un obstáculo (como un barco o una persona), este algoritmo sería capaz de evitarlo trazando una nueva ruta en tiempo real, y esto se consigue aun teniendo una carga computacional menor que una gran mayoría de métodos similares.

Para mostrar la diferencia entre el FM y el FM^2, el mejor ejemplo posible se encuentra en un artículo de divulgación [18] realizado por el tutor de este trabajo junto a dos compañeros más:

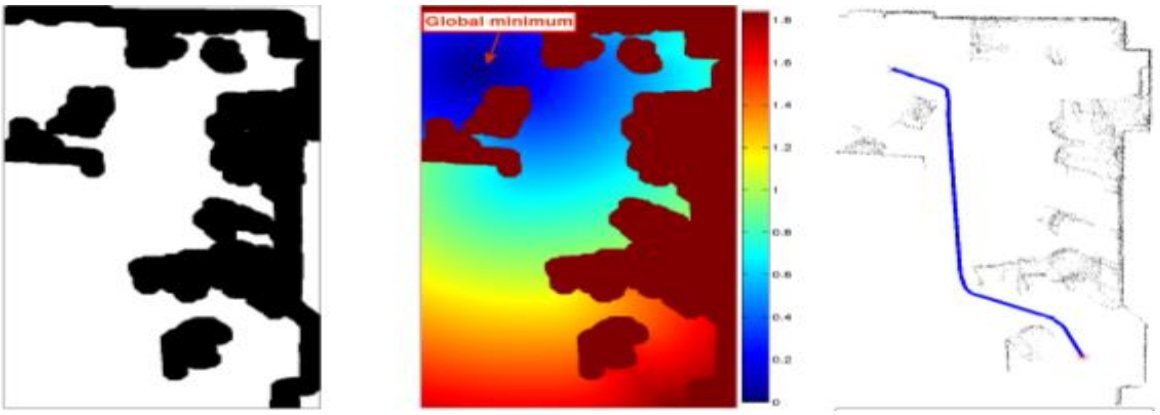


Figura 3.12: FM tradicional

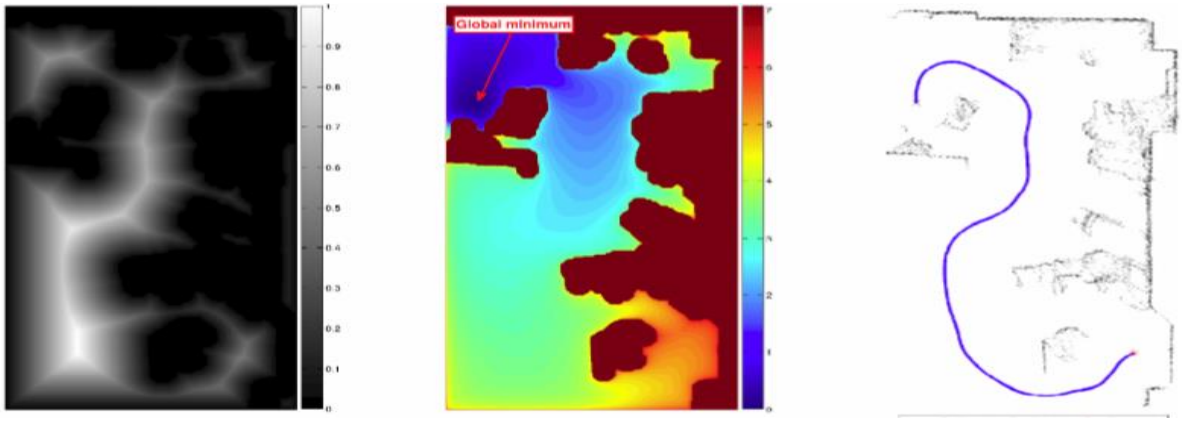


Figura 3.13: FM^2

En el FM tradicional tenemos el mapa en blanco y negro, el potencial de llegada, y finalmente la trayectoria entre el punto inicial y el final. En FM^2, tenemos a la izquierda el primer potencial W(x), del que obtenemos el nuevo potencial de llegada D(x), dando como resultado una trayectoria más suave y segura, alejada de los obstáculos.

La diferencia se ve intuitivamente en que la imagen de la izquierda, en FM clásico, es totalmente o blanca o negra por lo que cualquier punto que no sea negro es válido para la trayectoria, provocando que se seleccione aquel más cercano para que sea el camino más rápido. Sin embargo, en FM^2, hay una escala de grises según nos alejamos de las barreras, lo que hace que la trayectoria no pueda pasar tan pegadas a éstas.

Este es el motivo por el cual en el ejemplo de la introducción, *Room*, en vez de utilizar este potencial repulsivo:

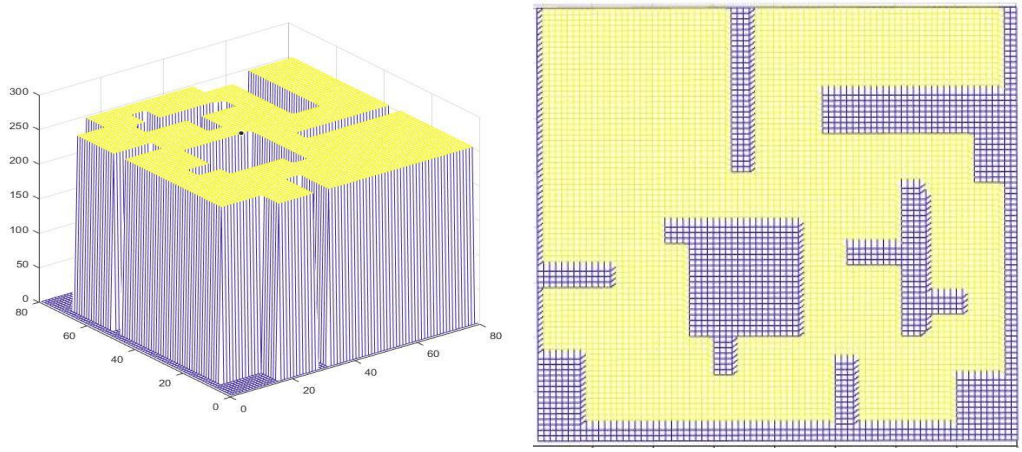


Figura 3.13: Primer potencial con FM en Room

Es mejor utilizar, y se utiliza, este otro:

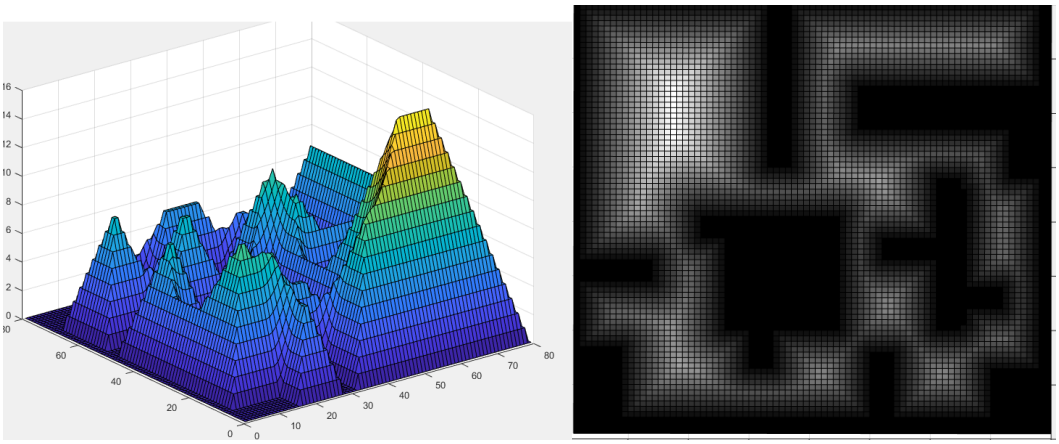


Figura 3.14: Primer potencial con FM^2 en Room

Es decir, dada una imagen en blanco y negro, aplicamos el Fast Marching obteniendo el primer potencial, W(x), que funciona a modo de mapa de velocidades o de índice de refracción, representando la distancia al obstáculo más cercano y teniendo un color más claro cuanto más alejado esté.

Se vuelve a aplicar este método sobre el mapa previo, W(x), y obtenemos un segundo potencial, D(x), que añade un tercer eje a W(x) representando el tiempo de llegada y se propagará desde el punto inicial hasta el destino, siendo este el mínimo local. A partir de ahora se empleará indistintamente la nomenclatura FM y FM^2.

Resumiendo, el proceso sería:

1. Partimos del W_0, un mapa binario que representa en negro los obstáculos y en blanco los espacios libres; en nuestro objetivo final representarán la tierra y el mar, respectivamente.
2. El primer potencial, W(x), se calcula aplicando el algoritmo del Fast Marching al mapa binario W_0. Este potencial, también llamado potencial repulsivo, funciona a modo de perfil o mapa de velocidades, ya que representa la distancia al obstáculo más cercano y, por tanto, indica la velocidad máxima que se puede tener en tales puntos sin poner en riesgo la seguridad. Este potencial, según qué literatura, también se conoce como potencial de dificultad o lentitud.
3. Se vuelve a aplicar el FM sobre el potencial previo, W(x), y obtenemos un segundo potencial, D(x), también llamado potencial atractivo. Dicho potencial representa el tiempo de llegada y se propagará desde el punto inicial hasta el destino, siendo este el mínimo. Hay dos formas principales de representación, ambas útiles:
    - Puede representarse el tiempo de llegada según el color; de este modo, el azul representará tiempo de llegada breve (alrededor del origen) y el rojo significará un tiempo mayor. En este proyecto se denominará tiempo de llegada.
    - Otra alternativa es representar el tiempo de llegada como eje vertical, de modo que el origen y sus alrededores tienen una altura cero, mientras que esta aumenta según nos alejamos. Se llamará tiempo de llegada vertical.

4. Aplicaciones

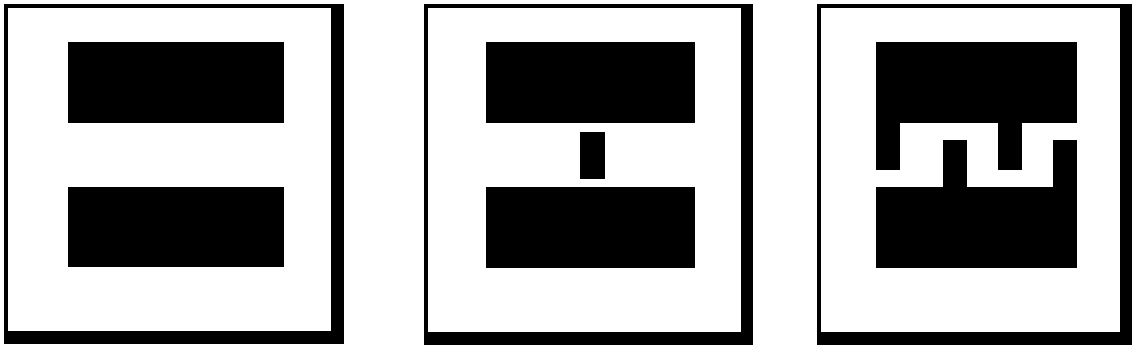
4.1. Primera fase

En este apartado haré un recorrido por los más importantes desafíos que he encontrado desde que comencé el proyecto y las soluciones que finalmente se utilizaron para resolver las dificultades partiendo desde ejemplos sencillos creados por mí o mi tutor hasta mapas reales.

Me centraré en los problemas realmente relacionados con el FM, aunque he de comentar que otros ligeros inconvenientes retrasaron en gran medida el desarrollo del proyecto: por ejemplo, varias funciones imprescindibles del código del FM no funcionaban, y continuó siendo así hasta que, tras investigar qué ocurría, descargué el compilador MinGW-W64, y tras compilar todo de nuevo, comenzó a funcionar mejor. Como este caso, me encontré con muchos obstáculos que impedían el correcto funcionamiento del algoritmo y problemas técnicos con Matlab y el código que retrasaron el trabajo varias veces durante semanas; pero me limitaré a nombrar solo aquellas incidencias que se basan en el FM y en su desarrollo.

4.1.1 Pasillo

Voy a comenzar con un ejemplo simple pero muy interesante ya que permite familiarizarnos con el método analizando cómo cambia el tiempo de llegada y los potenciales al variar el terreno. La imagen es muy simple, similar a la de Room: una habitación con dos grandes obstáculos entre los que iré añadiendo más barreras. Estas son las tres fases con las que voy a trabajar:



a)Pasillo 1 b)Pasillo 2 c)Pasillo 3

Figura 4.1: Representación de los pasillos

El origen y el destino son elegidos de tal modo que el camino atraviese el pasillo central. Es interesante notar que la parte blanca del lado superior es más estrecha que la del inferior, lo que provocará diferencias como vamos a ver.

En el caso a) obtenemos la siguiente trayectoria y tiempo de llegada:

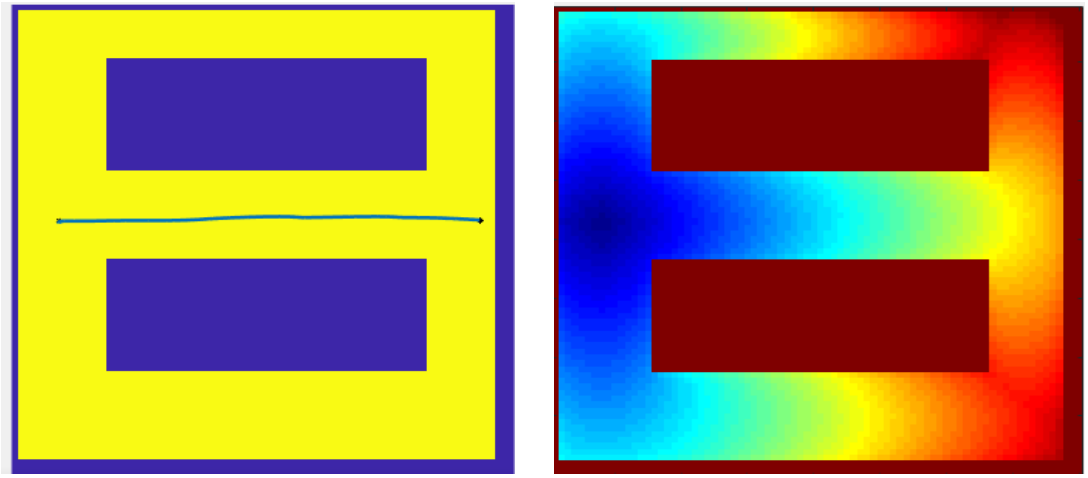


Figura 4.2: Trayectoria y tiempo de llegada en pasillo a)

Es importante observar que la trayectoria no es del todo recta debido a que hay corrientes, lo que será explicado más adelante. Color azul significa tiempo de llegada breve, y rojo significa que se tarda más en llegar. Ahora vamos a comparar el tiempo de llegada de a) con los de los otros dos casos para ver cómo evoluciona:

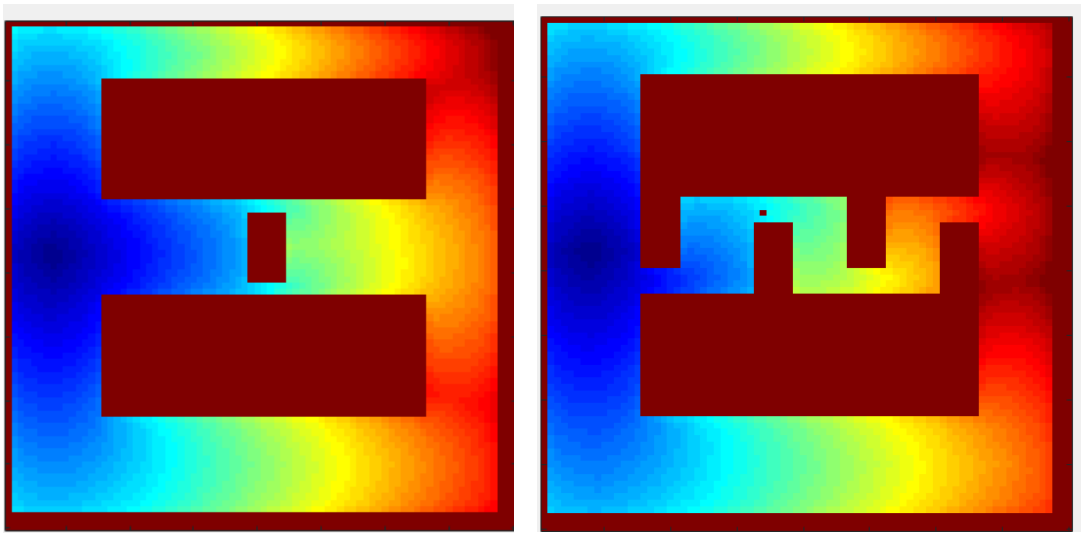


Figura 4.3: Tiempo de llegada en pasillos b) y c)

Como era de esperar, los obstáculos ralentizan el camino central mientras que los caminos de ambos laterales permanecen igual tal y como muestra la evolución de los colores. En el caso b) se puede notar que justo después del obstáculo comienza a aparecer antes el color amarillo por el centro que por los laterales, ya que atravesarlo pegado a la pared sería más rápido que esquivar el rectángulo para volver al centro. En el caso c) el retraso es aún mayor, llegando a la zona roja antes incluso de abandonar el pasillo.

Ahora vamos a analizar el resultado de aplicar Fast Marching Square obteniendo el primer potencial (o potencial repulsivo), que es muy útil pues muestra lo favorable que es una coordenada del mapa. En el caso a) obtenemos lo siguiente:

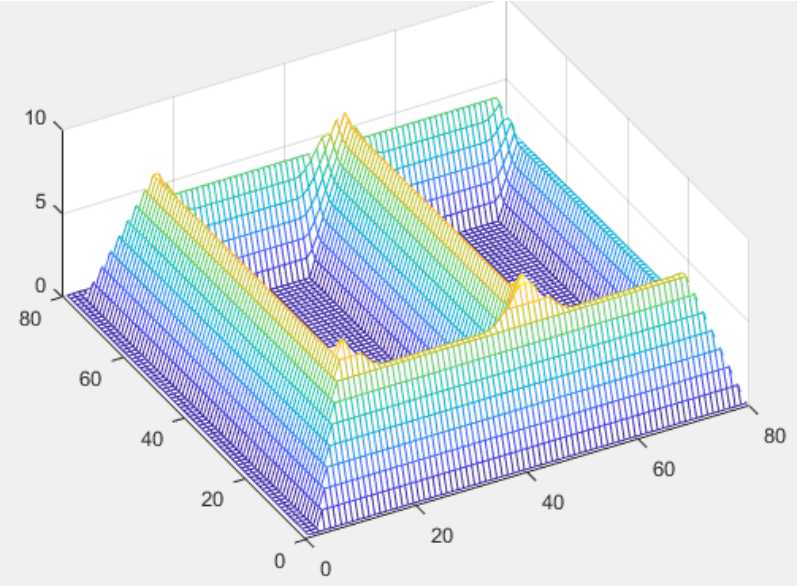


Figura 4.4: Primer potencial en pasillo a)

Este potencial muestra el valor de cada coordenada: cuanto más alto sea, mejor. La altura es proporcional a la distancia que guarda respecto al obstáculo más cercano, por ello el lado superior del mapa (el derecho en el la figura 4.4) tiene una altura menor al ser más estrecho. Ahora comprobemos como sale este potencial en los otros dos ejemplos:

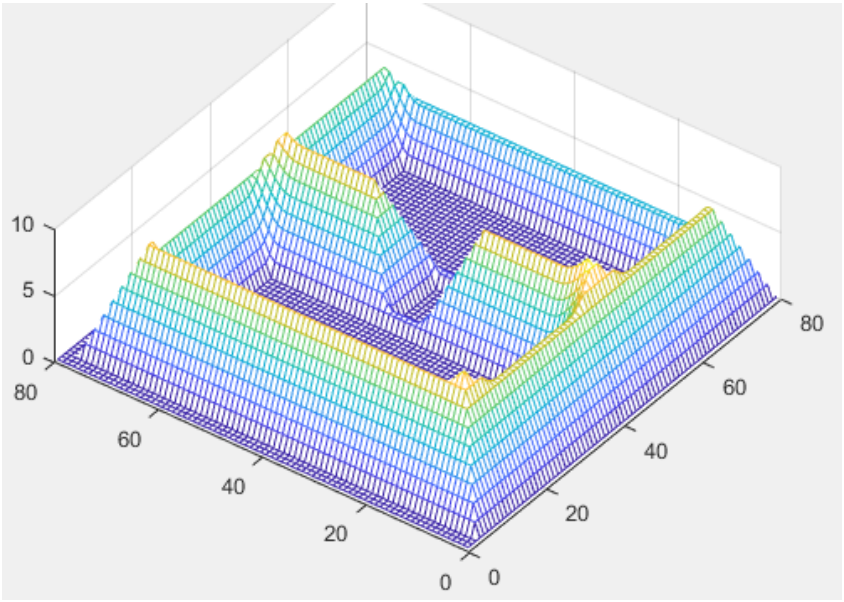


Figura 4.5: Primer potencial en pasillo b)

En el caso b) hay una gran depresión en el camino central debido a que el obstáculo está en medio y la onda solo puede expandirse por los huecos que hay entre este y la pared; por ello solo se ven dos hileras muy pequeñas por los laterales del camino central.

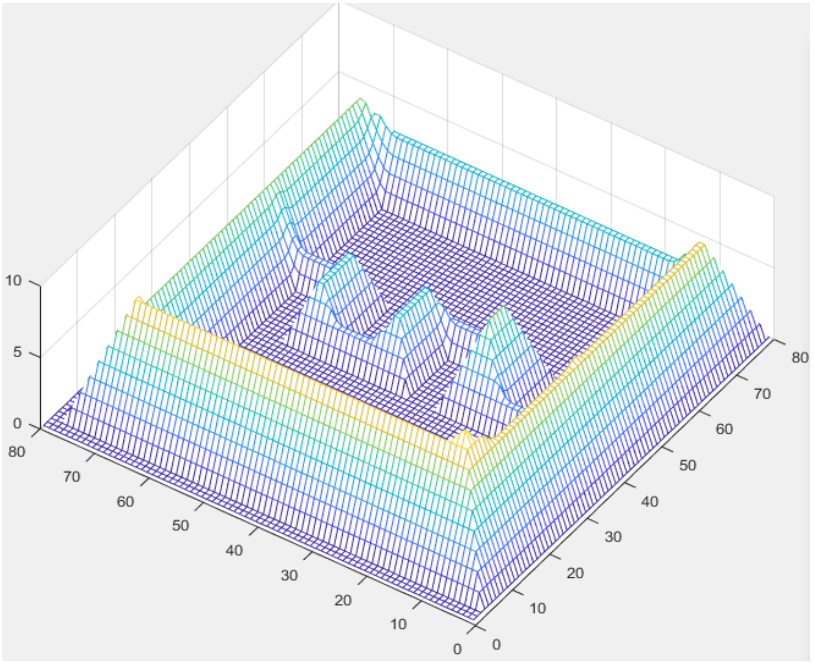


Figura 4.6: Primer potencial en pasillo c)

En este caso las alturas del camino central no son tan pequeñas ya que los obstáculos, al estar pegados a la pared -en vez de en mitad del camino-, guardan una mayor distancia respecto a la pared de enfrente, concretamente el doble. Es decir, en este caso estas coordenadas son más favorables que las del caso b) ya que guardan mayor distancia y por lo tanto son más seguras; sin embargo, el trayecto c) es más lento que en b) como se ve en el tiempo de llegada ya que la trayectoria tiene que perder tiempo haciendo curvas.

Esto nos muestra algo importante y es que el hecho de que un camino sea más favorable que otro atendiendo a su primer potencial no significa que sea el óptimo, ya que esto solo se puede comprobar atendiendo al tiempo de llegada.

Por último, vamos a analizar el tiempo de llegada poniéndolo como eje vertical, para verlo en tres dimensiones en vez de a través del color. En los casos a) y c) obtenemos los siguientes tiempos de llegada, visto desde dos perspectivas diferentes:

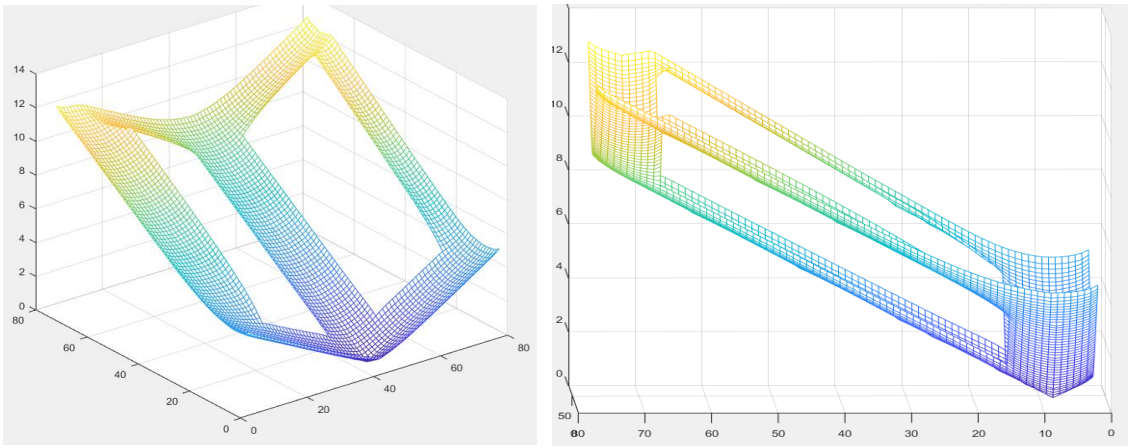


Figura 4.7: Tiempo de llegada vertical en pasillo a)

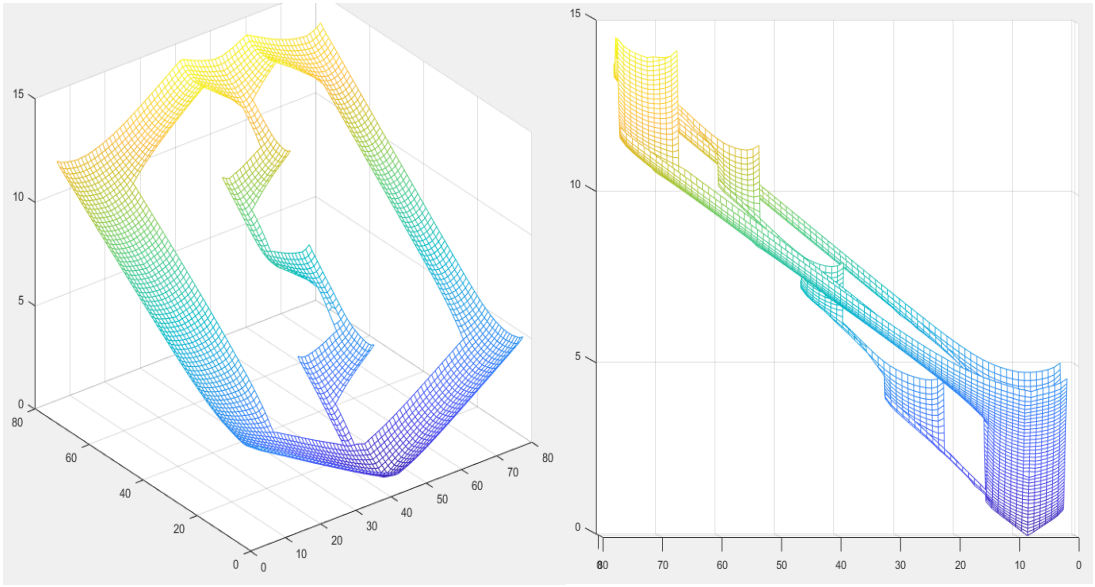


Figura 4.8: Tiempo de llegada vertical en pasillo c)

Hay que entender que en el FM, primero se elige el punto de origen, que es el punto más bajo pues es desde el cual se propaga la onda. La forma más sencilla para entender cómo funciona este potencial es imaginar que se coloca una canica en cualquier punto de la figura: esta realizará el camino más breve, es decir aquel con mayor pendiente (mayor velocidad, menor tiempo) desde el punto donde la coloquemos hasta el origen.

Puede parecer contraintuitivo que el origen en este ejemplo sea el punto donde termina la canica, pero realmente el potencial comienza a expandirse desde dicha coordenada por lo que es el modo de funcionamiento; y es similar al comportamiento de, por ejemplo, la energía potencial: con un sistema de referencia simple, un cuerpo tiene mayor energía potencial cuanta más distancia guarde respecto al suelo, del mismo modo, aquí los puntos tienen mayor potencial al aumentar la distancia respecto al origen.

Se ve a simple vista el efecto de los obstáculos: estos curvan el espacio a su alrededor de un modo similar al afecto gravitatorio de cuerpos masivos en la teoría de la relatividad. Es decir, los obstáculos alteran el tiempo de llegada de las coordendas vecinas haciendo que aumente, esto es lo que se traduce como colores más cercanos al amarillo o rojo en la figura 4.3.

Cada vez que hay una barrera la cuesta se inclina hacia arriba mientras que el origen es el punto más bajo (mínimo global). De hecho, se puede comprobar que donde más se curva -hacia arriba- la bajada es precisamente en las esquinas entre obstáculo y pared, lo que es de sentido común ya que si recorres el pasillo yendo de esquina a esquina estás realizando el trayecto más largo posible.

4.1.2 Caverna

En este caso comenzaré a mostrar las incidencias más comunes que he ido encontrado al experimentar con el código.

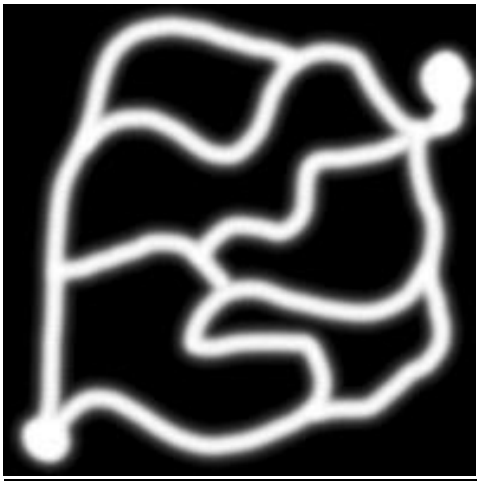


Figura 4.9: Imagen de Caverna

Supongamos que deseamos ir de la esquina inferior izquierda a la salida, el trayecto que obtenemos con el software es:

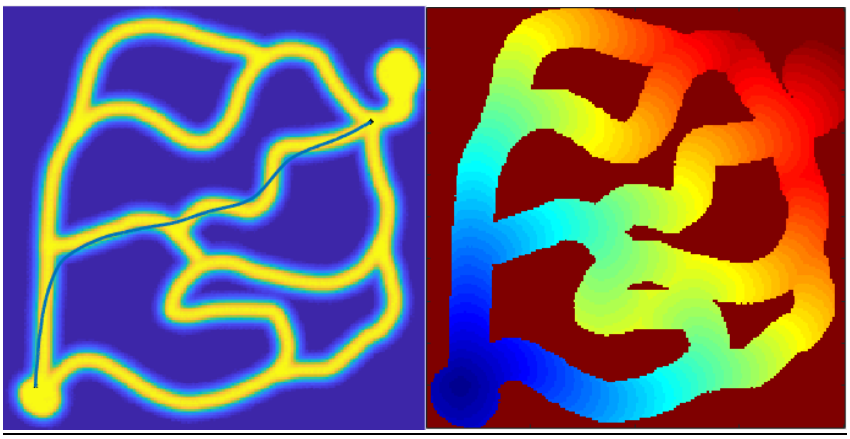


Figura 4.10: Trayecto y tiempo de llegada en caverna

El tiempo de llegada ha funcionado sin problemas, sin embargo el trayecto trazado no es convincente ya que pasa excesivamente cerca de las esquinas, y de hecho en algunas curvas da la sensación de que atraviesa la zona prohibida. El motivo de esto, como casi siempre, se encuentra en la imagen de partida: esta no salta del blanco al negro de forma drástica, sino que tiene un cierto halo grisáceo entre medias, lo que a la hora de la verdad se traduce en que FM cree que puede atravesar esa zona al no ser completamente negra. La solución para ello es sencilla: hay que buscar, o diseñar, imágenes que estén totalmente en blanco o en negro, sin intermedios.

4.1.2 Serbia

En este caso, partimos de una imagen del territorio serbio [19] como superficie a evitar:

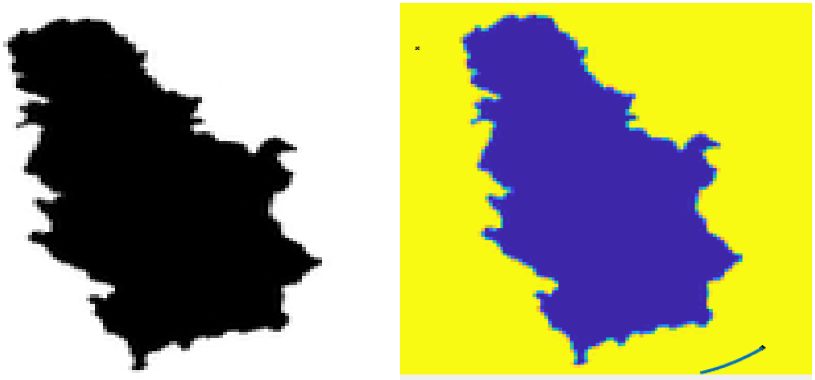


Figura 4.11: Imagen y trayectoria en Serbia

El punto inicial del trayecto se encuentra en la esquina superior izquierda, y el destino en la esquina inferior derecha. Como se puede observar, el *path* escogido directamente se sale de la imagen, estando evidentemente mal. Ahora bien, ¿por qué sucede esto?

Esto se debe a que los bordes no son negros, por lo que la velocidad es "infinita" en ellos y por tanto el algoritmo elige a estos como camino más rápido, dando la apariencia de que realmente sale de la imagen. Se puede solucionar fácilmente si añadimos unos simples bordes negros, a modo de marco, a la imagen:

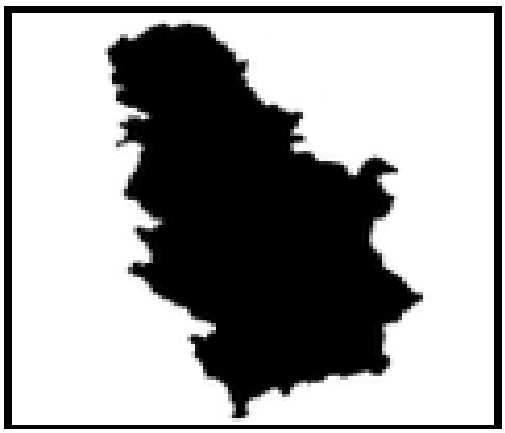


Figura 4.12: Serbia con bordes negros

En este caso obtenemos los siguientes resultados:

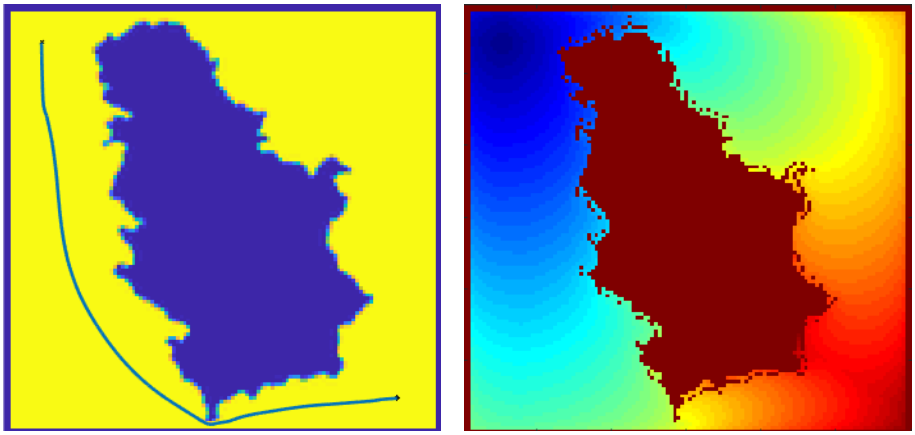


Figura 4.13- Trayectoria y tiempo de llegada en Serbia con bordes negros

Esto se ve con más claridad si comparamos el primer potencial (o perfil de velocidad) de ambas imágenes; en la primera imagen, que no tenía bordes negros, obtenemos esto:

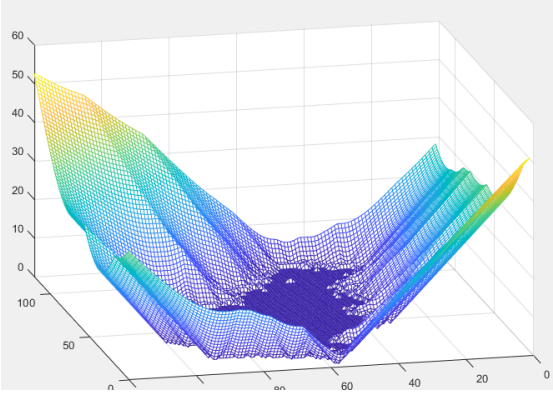


Figura 4.14: Primer potencial en Serbia sin marco

Es decir, la velocidad tiende al máximo en los extremos, lo que es un problema ya que la trayectoria tratará de pasar por estos alejándose lo máximo posible de la imagen. Por otro lado, al añadir marco el marco negro obtenemos lo siguiente:

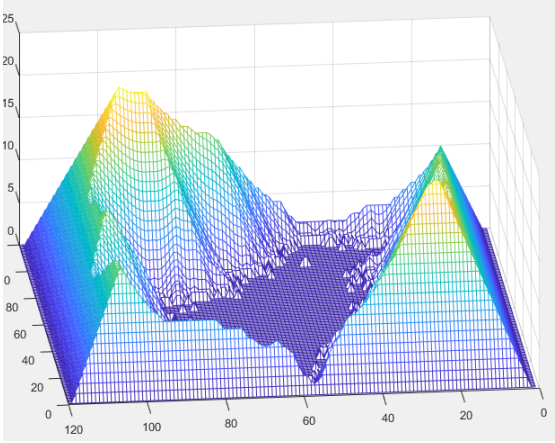


Figura 4.15: Primer potencial en Serbia con marco

Como se ve, aquí la velocidad (altura) va creciendo según nos alejamos de ambos bordes negros (el del marco de la imagen, y el del territorio serbio) y así obtenemos lo que deseamos: que el camino óptimo esté entre medias de ambos, es decir, la velocidad tiene que ser nula tanto en el marco del mapa como en el territorio negro.

En el centro del eje horizontal (cerca de la coordenada 60) es visible una gran depresión en la altura, esto se debe a que el hueco entre el borde en forma de pico de Serbia y el marco negro es muy estrecho, por lo que el algoritmo penaliza este fragmento del trayecto ya que el objetivo es pasar lo más alejado posible de los límites. Sin embargo no siempre basta con añadir un marco, puede haber otros problemas irresolubles como veremos en el ejemplo *mapamundi.*

Sin embargo, aunque ahora encuentra el trayecto, el resultado sigue sin convencer del todo ya que en el tiempo de llegada (figura derecha en 4.13) el borde del territorio aparece mal trazado. ¿Por qué? La explicación la encontramos, de nuevo, en la imagen original en blanco y negro; por un lado, esta tiene demasiados bordes, demasiadas irregularidades y píxeles sueltos como se puede apreciar en la siguiente imagen:

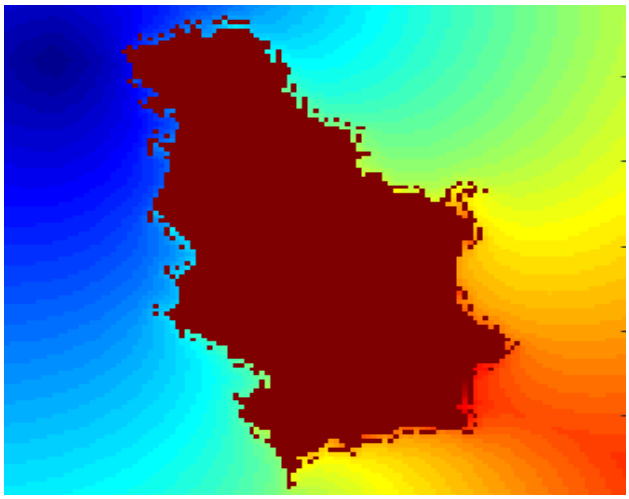


Figura 4.16: Tiempo de llegada en Serbia

Como se ve, el perfil del territorio no es uniforme, sino que aparecen puntos nuevos alrededor de toda la superficie, y esto es problemático ya que complica en exceso el proceso iterativo y matemático que tiene este método detrás, provocando que no sea capaz de dar una respuesta coherente ya que tienden al infinito los puntos intermedios, en vez de ser blancos o negros.

En las siguientes imágenes se ve con mayor claridad:

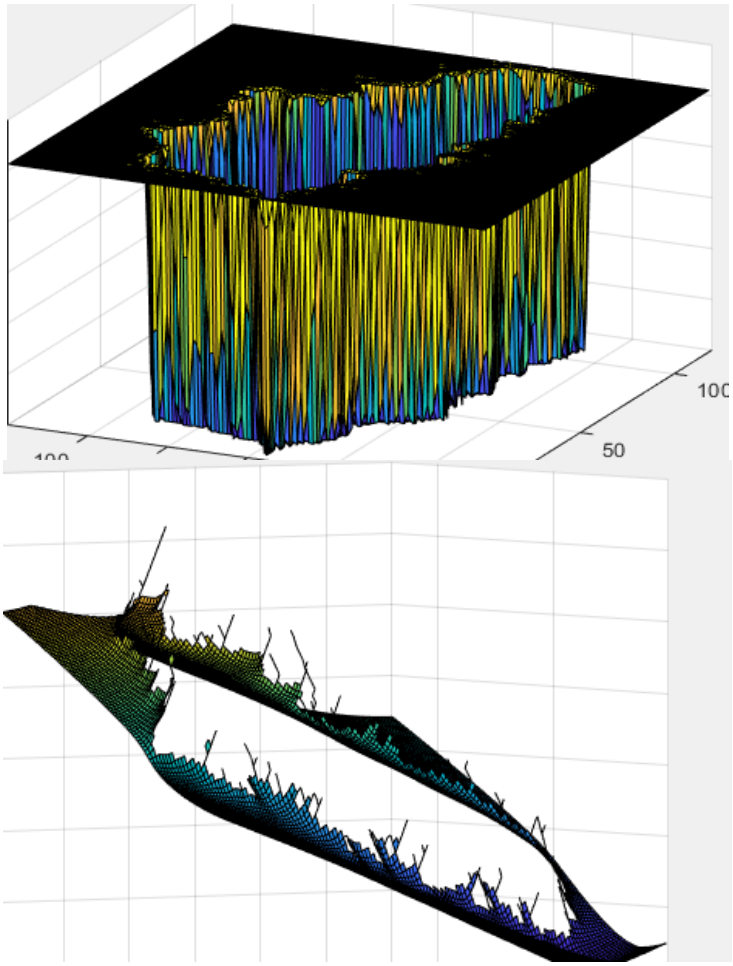


Figura 4.17: W_0 y tiempo de llegada vertical en Serbia

La W_0 ya no salta de 1 a 0 directamente, sino que tiene valores intermedios en demasía; del mismo modo, el tiempo de llegada ya no es siempre descendente, sino que salen unas ciertas dendritas o ramificaciones en sentido contrario provocando que el método sea incapaz de resolver correctamente este caso.

Como se ha dicho, el problema de esto se debe a que la superficie negra no tiene unos límites simples y claros, sino que tiene bordes muy irregulares que a la hora de calcular se convierten en píxeles que no están bien definidos.

Una solución para evitar esto es utilizar imágenes con mayor resolución, pero esto implica añadir una mayor carga computacional y el código solo ha funcionado realmente bien con imágenes comprendidas entre 100x100 y 200x200 píxeles por lo que una solución más eficaz es buscar imágenes con bordes bien definidos y regulares y con el menor número de picos posibles, como el ejemplo de *Room* donde todas las paredes son rectas. Evidentemente, es sencillo dibujar una habitación con paredes rectas mientras que es casi imposible encontrar un mapa así, por lo que hay que ser cuidadoso y probar con un gran número imágenes hasta encontrar una que funcione.

4.1.3 Mapamundi

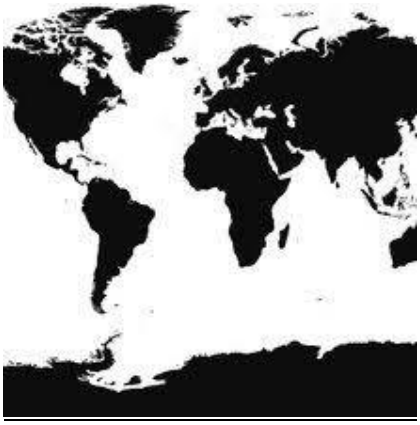


Figura 4.18: Imagen Mapamundi

Empleando ya un mapa en el que podemos simular viajes navales, obtenemos el siguiente trayecto:

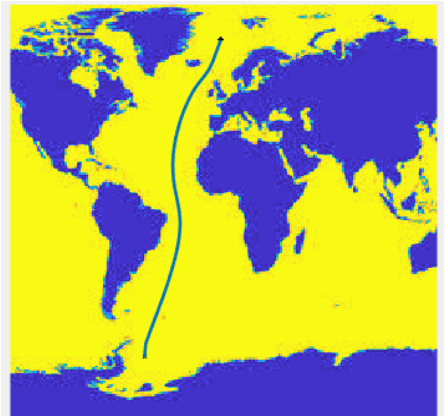


Figura 4.19: Trayectoria en Mapamundi

En esta imagen parece que se obtiene un buen camino pero no es así ya que siempre hay que revisar el tiempo de llegada, y en él encontramos lo siguiente:

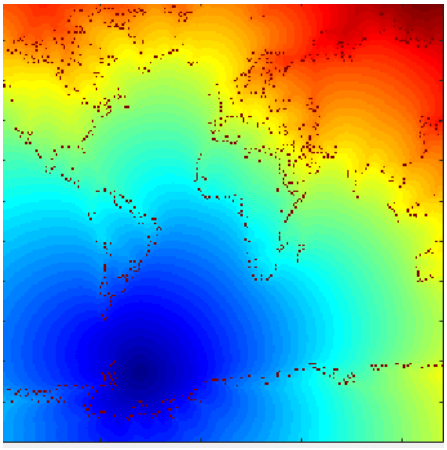


Figura 4.20: Tiempo de llegada en Mapamundi

Como vemos, el tiempo de llegada ignora la superficie actuando prácticamente como si no hubiese tierra (por ello la onda se expande casi en líneas isobaras respecto del origen). Esto se puede comprobar si elijo como destino un punto dentro del continente, que debería ser evitado al ser negro:

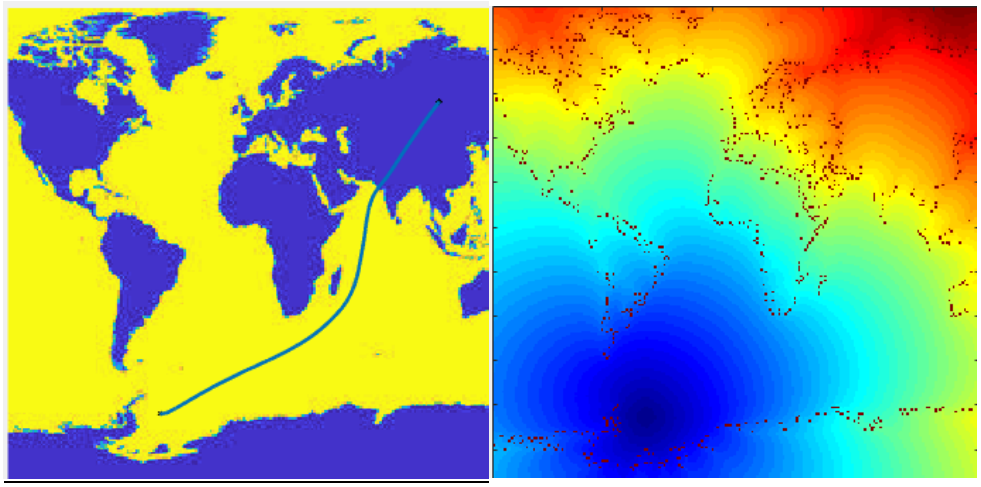


Figura 4.21 Trayectoria y tiempo de llegada en Mapamundi

Si el algoritmo funcionase bien con esta imagen, al ejecutar el código debería dar un error ya que es imposible que la onda alcance dicho destino en ninguna iteración, pero esto no ocurre así. ¿A qué se debe esto? De nuevo, es debido a la calidad y características de la imagen original.

En un primer análisis podemos pensar que, al igual que en el ejemplo de Serbia, se solucionaría añadiendo un marco negro de forma artificial, pero el problema persiste:

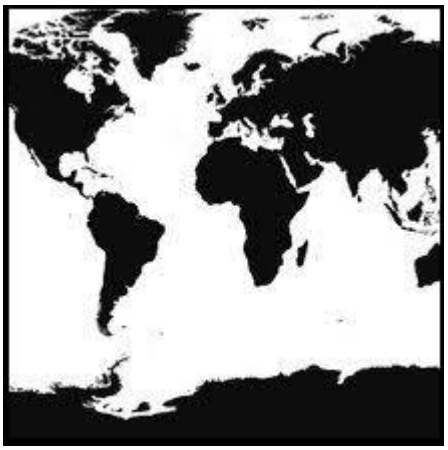


Figura 4.22: Mapamundi con marco

Sobre el mapa de mapamundi con marco, volvemos a elegir un punto de destino dentro del continete, y el resultado vuelve a ser erróneo:

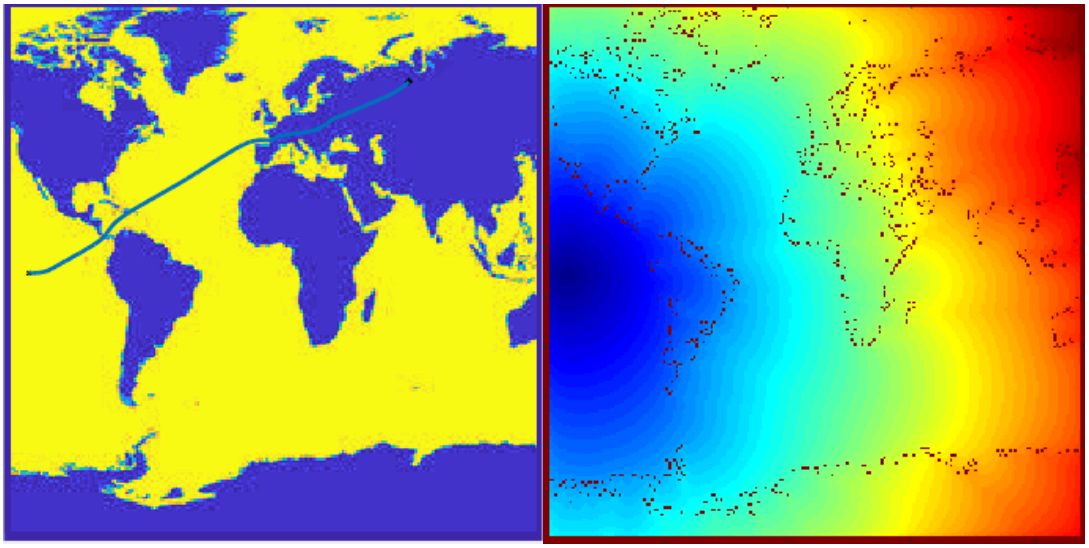


Figura 4.23: Trayectoria y tiempo de llegada en Mapamundi con marco.

Si vemos el primer potencial en la figura 4.24, no es todo negro y blanco, sino que hay casi infinitos puntos por los que se puede pasar, siendo como agujeros. Esto genera que el trayecto pueda cruzar entre estas zonas que deberían estar prohibidas, debido principalmente a que el negro de las imágenes no es negro puro (valor 0) sino que tiene un valor mínimamente superior (mínimamente grisáceo) haciendo que la onda pueda pasar entre ellos.

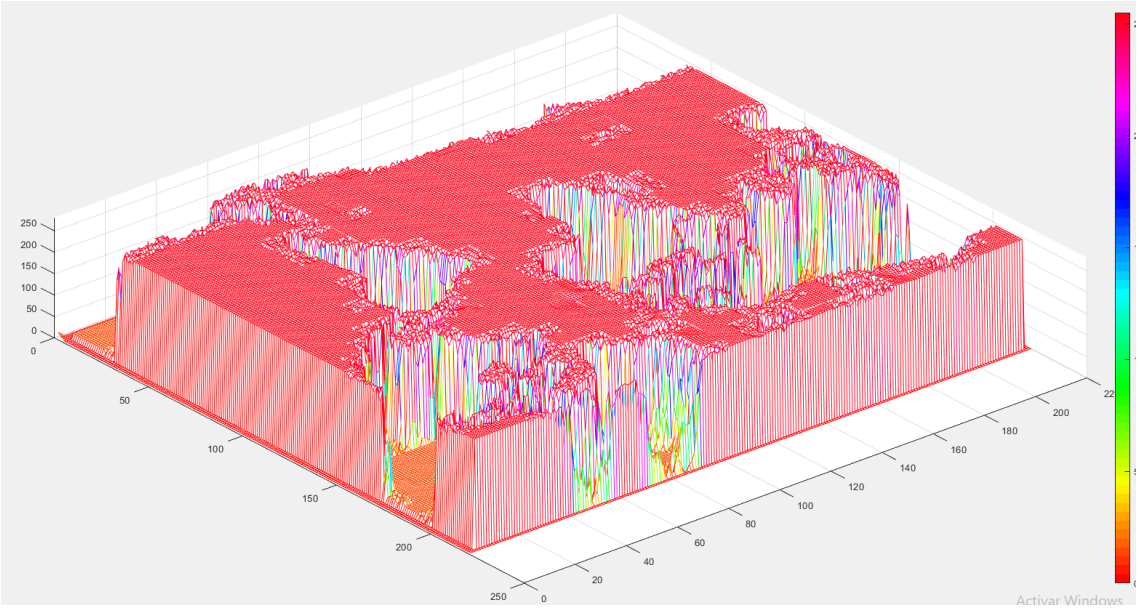


Figura 4.24: Primer potencial en Mapamundi

Sin embargo, usando otra imagen que también representa un mapamundi, sí conseguimos que funcione, como veremos en el siguiente ejemplo.

4.1.4 Mapamundi V2

Utilizando otra imagen de Mapamundi da un resultado mejor pero, aunque funcione, el resultado no es del todo correcto ya que en el tiempo de llegada vemos que se reducen partes de las zonas negras (como la brújula), recortando los bordes e incluso haciendo que desaparezcan islas. Es decir, ocurre el mismo problema que en el caso anterior, solo que en el caso previo directamente desaparecía toda la zona terrestre mientras que aquí desaparecen solo las partes más externas: esto se debe a que la geometría es compleja, por lo que tiene que hacer demasiados cálculos y acaba funcionando de forma errónea.

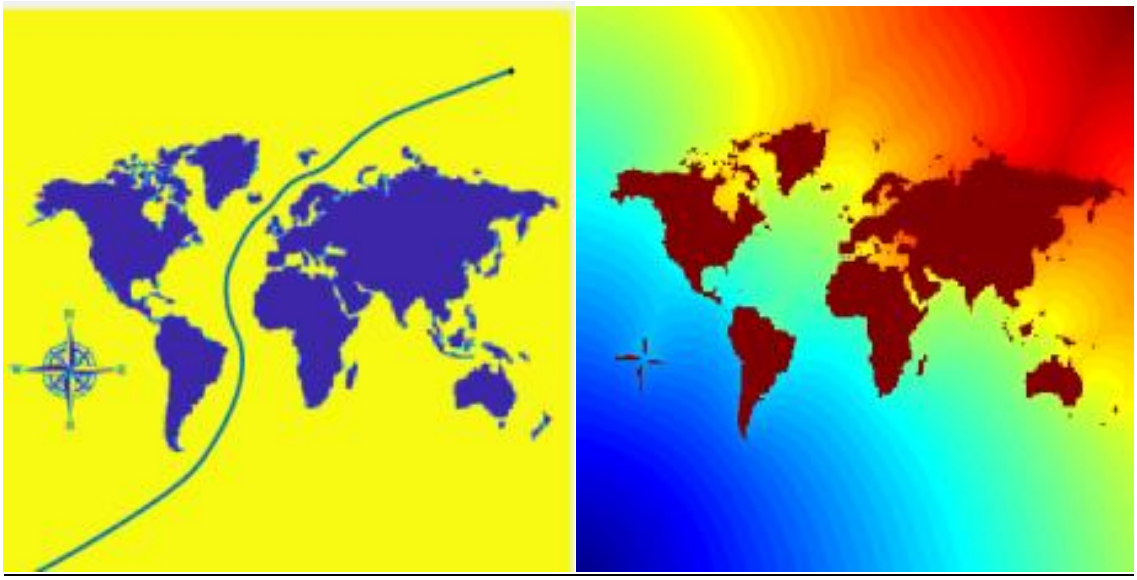


Figura 4.25: Trayecto y tiempo de llegada en Mapamundi 2

4.1.5 Laberinto

Ahora utilizaremos el ejemplo de un laberinto para mostrar que el FM siempre encuentra la solución en el caso de que haya alguna posible; y la relevancia del tiempo de llegada. Supongamos que tenemos el laberinto siguiente y que queremos ir desde el comienzo, en el lado izquierdo, hasta el final en el derecho:

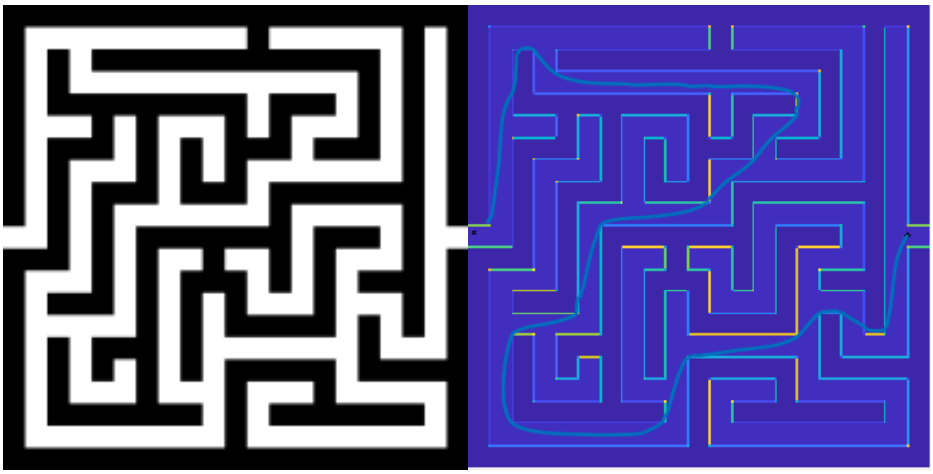


Figura 4.26: Trayectoria en Laberinto

Se observa que se ha encontrado el único trayecto posible y que el proceso iterativo no ha dado problemas en este caso ya que, aunque ha tenido que hacer muchas repeticiones por el diseño del circuito, las paredes son rectas y están bien definidas evitando así la aparición de incidencias; y desde luego los colores son blanco y negro puros.

Es muy interesante observar el tiempo de llegada, sobre todo al ponerlo en el eje vertical, ya que traza directamente la solución al laberinto:

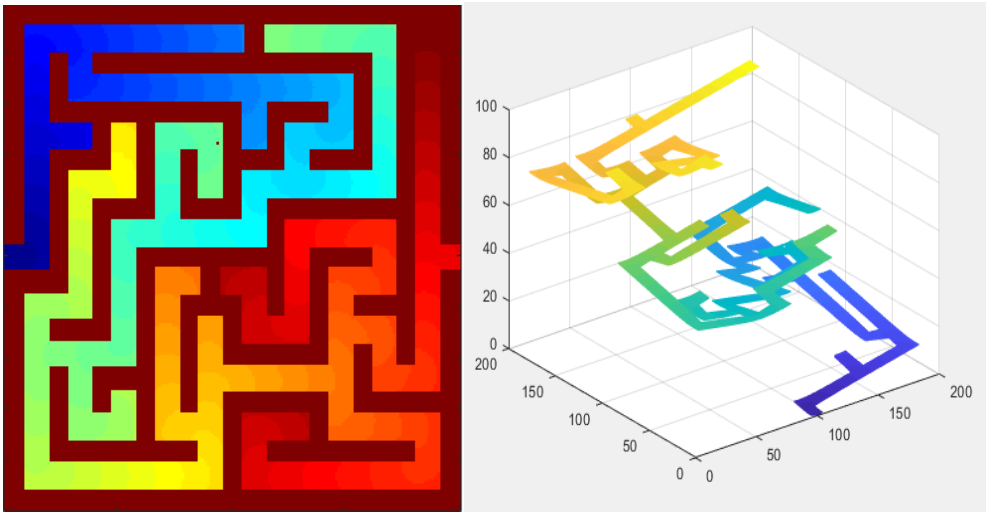


Figura 4.27: Tiempo de llegada y tiempo de llegada vertical en Laberinto

4.2 Segunda fase: funcionamiento de las corrientes

4.2.1 Ejemplo 1

Para introducir las corrientes, las cuales funcionan a modo de campo vectorial, partimos de un ejemplo básico. Es interesante comprobar el primer potencial de la siguiente imagen, que forma una estructura perfectamente piramidal al ser un cuadrado perfecto, por lo que los puntos se alejan simétricamente respecto de las 4 paredes:

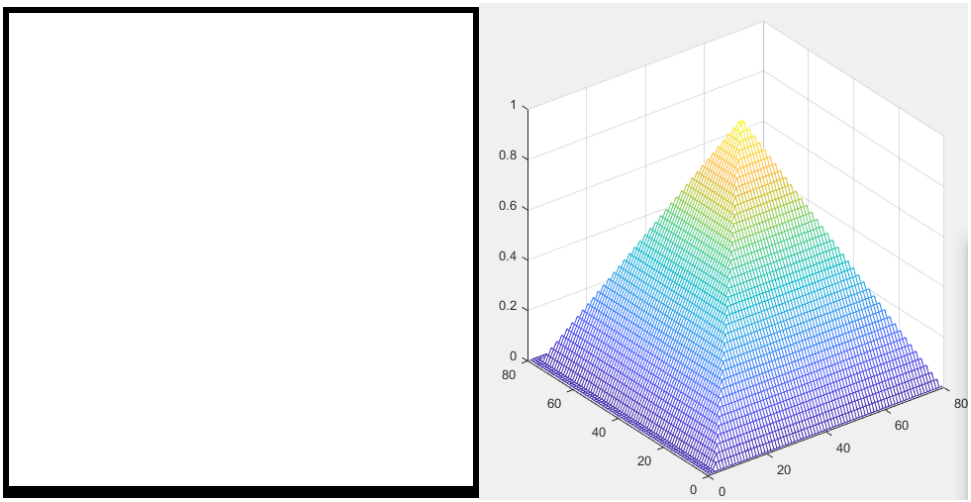


Figura 4.28: Imagen y primer potencial con corrientes ejemplo 1

A continuación voy a representar las trayectorias en tres casos: a la izquierda, con una corriente horizontal negativa; en el centro, con corrientes nulas; y a la derecha, con una horizontal positiva, tal y como muestran las flechas.

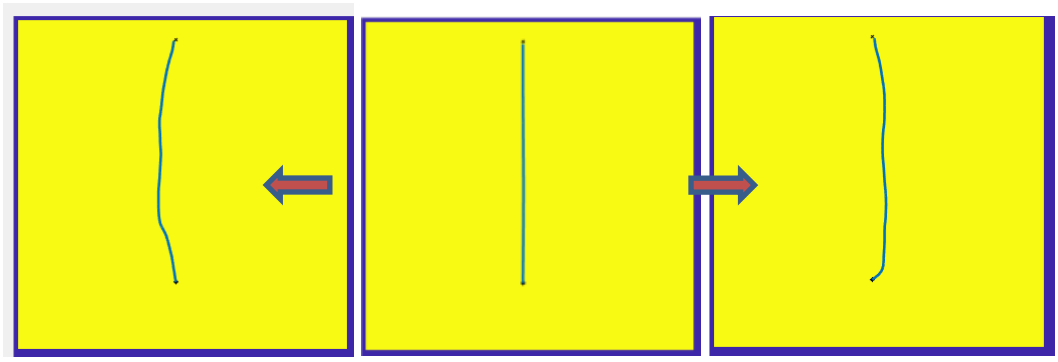


Figura 4.29: Trayectorias con corrientes en ejemplo 1

El efecto es evidente: la corriente empuja el obstáculo en el sentido de esta como se ve en los casos de la derecha y la izquierda; esto es así mientras no haya obstáculos, con ellos el funcionamiento se complica como veremos en los siguientes ejemplos. En el caso del centro, al no tener corriente, la trayectoria es una recta.

Merece la pena comparar los tiempos de llegada de los tres casos ya que nos da una visión muy intuitiva y útil de cómo afectan las corrientes al segundo potencial; los tres casos están ordenados en el mismo sentido que antes:

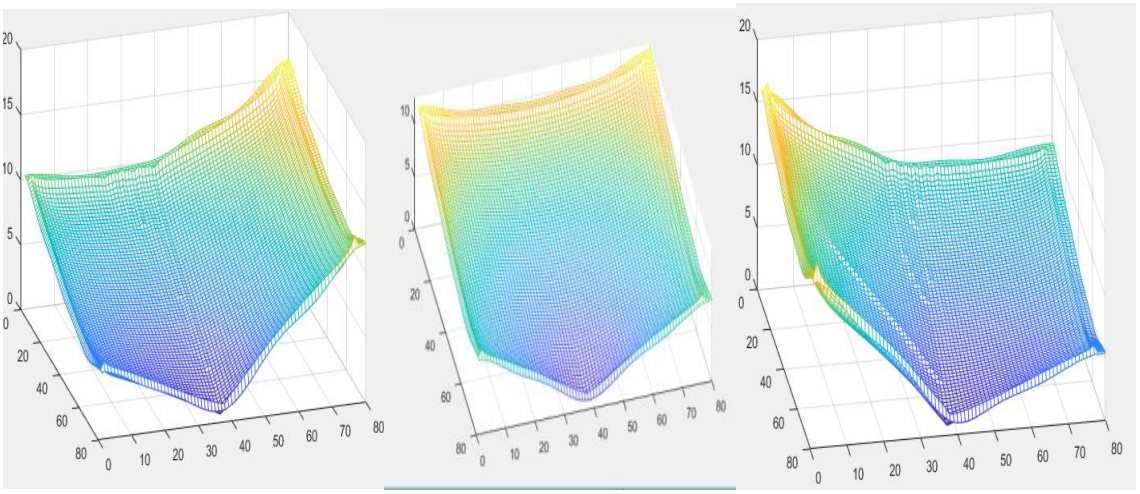


Figura 4.30: Tiempo de llegada con corrientes en ejemplo 1

En el caso central, al no tener corriente y ser una imagen simétrica, el tiempo de llegada crece desde el punto inicial de igual forma a lo largo de toda la superficie. En cambio, en el caso de la izquierda, al tener una corriente negativa en el eje horizontal, esta te empuja hacia la izquierda por lo que gastarías más tiempo si navegas por el flanco derecho (yendo a contracorriente), y por ello la trayectoria salía por el flanco contrario. Por ello se ve claramente que el lado derecho se eleva con mayor intensidad que el contrario, significando que el tiempo de llegada a esos puntos es mayor que el que encontramos en las coordenadas simétricas del otro flanco. En el caso derecho es el mismo ejemplo solo que al tener el vector de corriente con sentido hacia la derecha, el resultado es exactamente el contrario.

4.2.2 Ejemplo 2

Ahora vamos a utilizar un ejemplo algo más complejo, teniendo el mismo punto de inicio y fin, y probando con cuatro configuraciones distintas de corrientes, es decir, teniendo la corriente horizontal en X (FX) como 1 o -1, y lo mismo en la dirección vertical Y. Este es el resultado:

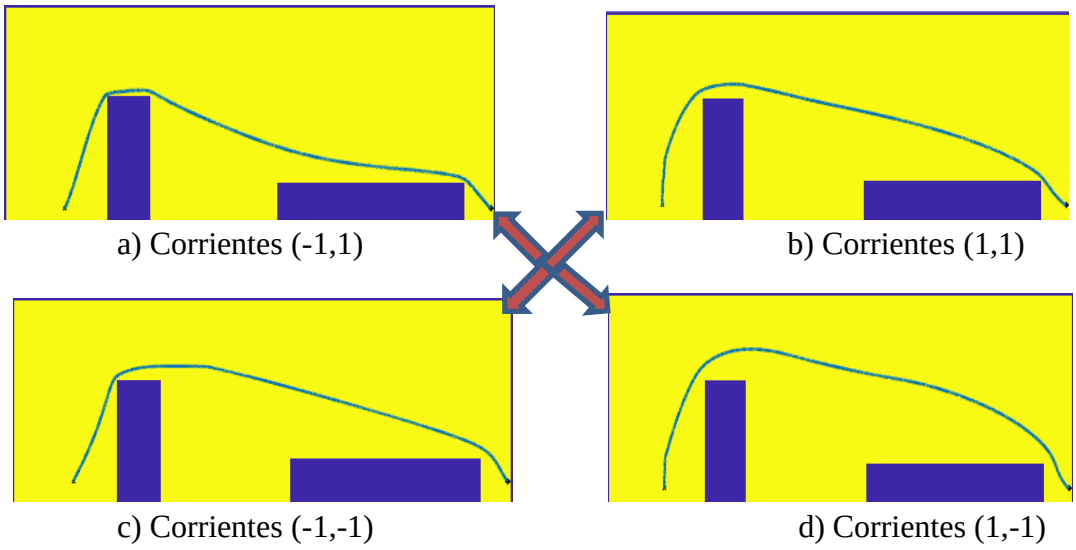


Figura 4.31: Trayectorias según corriente ejemplo 2

Se han dispuesto las 4 imágenes de modo que la corriente varíe en el sentido de los cuadrantes en ejes cartesianos; las flechas indican el sentido de la corriente en cada caso para hacerlo más visual.

El funcionamiento de la corriente es el siguiente: si es favorable en la dirección de choque potencial con el obstáculo, genera una distancia de seguridad mayor, como en el caso b) y d); en cambio, si la corriente empuja en dirección contraria al obstáculo, no hará falta una distancia de seguridad tan grande ya que el riesgo es menor pues el propio campo vectorial (la inercia de la fuerza del mar) te aleja del obstáculo, como vemos que ocurre en a).

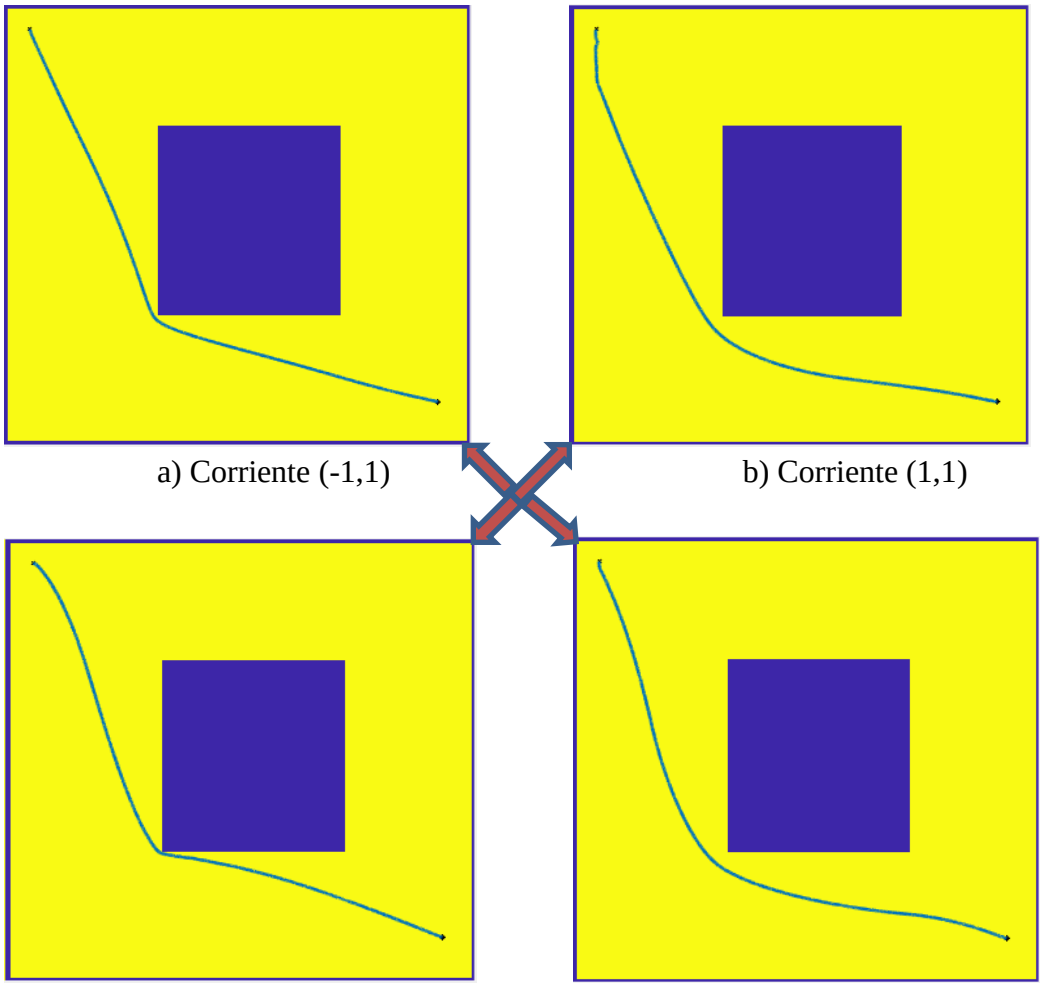
Es intuitivo: imaginemos una canoa que navega pegada a una orilla rocosa y la corriente le empuja contra esta, es evidente que la trayectoria y la fuerza de la canoa debe oponerse a la corriente para evitar colisionar.

Un valor negativo en X como los de los casos a) y c) provoca que el trayecto pase más cerca del primer obstáculo, ya que la corriente empuja hacia la izquierda mientras que el objeto se encuentra a la derecha del origen, por lo que se puede guardar menos distancia sin comprometer la seguridad.

Sumando ambos casos, es evidente que el ejemplo en el que más cerca se pasará es el a), ya que empuja a la corriente hacia la izquierda y hacia arriba, justo en la dirección opuesta al primer obstáculo.

Con el segundo obstáculo (horizontal) es similar: una corriente negativa en Y genera una mayor distancia para evitar una posible colisión, como se ve con total claridad en d).

Vamos a analizar de nuevo las corrientes y poner a prueba el funcionamiento teórico del ejemplo anterior, pero ahora con una trayectoria algo más compleja, ya que va desde la esquina superior izquierda hacia la inferior derecha, es decir, trazando un trayecto hacia abajo y hacia la derecha:



a) Corriente (-1,1) b) Corriente (1,1)

c) Corriente (-1,-1) d) Corriente (1,-1)

Figura 4.32 Trayectorias según corriente ejemplo 3

En el caso c), la corriente negativa en X genera un campo vectorial que empuja la corriente hacia la izquierda, teniendo en obstáculo a la derecha, por lo que se mantiene más cerca del vértice del cuadrado.

En el caso d), vemos que una corriente que empuja hacia abajo casi no afecta (solo desplaza ligeramente el camino hacia abajo) debido a que la trayectoria ya va hacia abajo por donde esta el objetivo, es decir, hay que diferenciar dos casos:

- Si navegas a contracorriente, una corriente contraria a tu trayectoria provoca que pases más cerca de los obstáculos (menor distancia de seguridad), como ocurre en el caso c) y a).
- Si navegas a favor de la corriente, se genera una mayor distancia ya que la corriente te empuja en el mismo sentido en el que tu navegas, como en el caso d).

Esto es aplicable en este ejemplo porque el obstáculo siempre está a la derecha de la trayectoria en los momentos críticos; sino, el comportamiento sería inverso o distinto, según su posición.

Ahora vamos a comprobar en el ejemplo de *Room* como, teniendo el mismo punto inicial y el mismo punto final, la trayectoria cambia variando únicamente las corrientes.

Como se puede ver en el código que esta sobre las siguientes imágenes, en ambos casos los puntos inicial y final son [14;25] y [73;33] respectivamente, lo único que cambia son los valores de FX y FY, ya que en el primer caso tenemos FX=FY=1, mientras que en el segundo caso FX=1 y FY=-2:

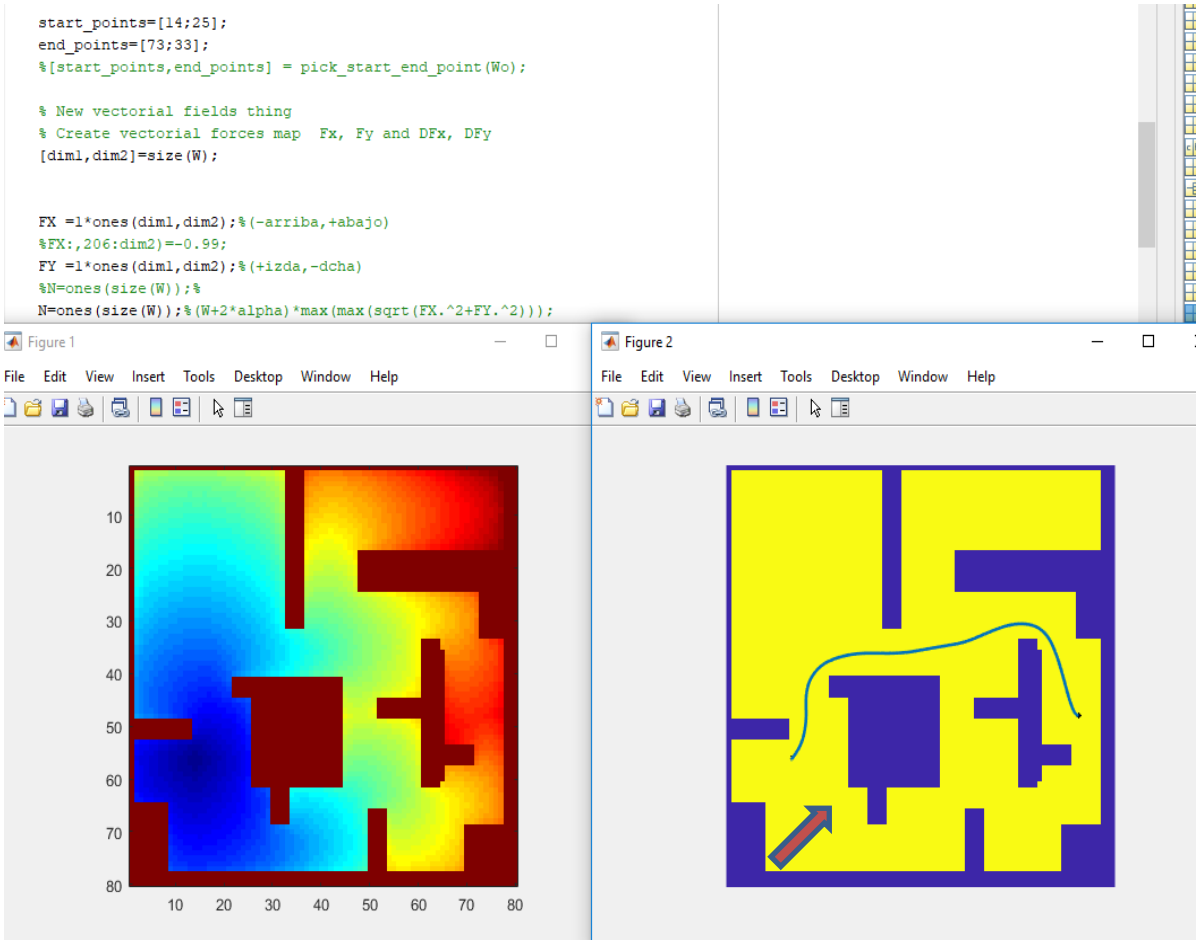


Figura 4.33: Room con corrientes (1,1)

```
start_points=[14;25];
end_points=[73;33];
%[start_points,end_points] = pick_start_end_point(Wo);

% New vectorial fields thing
% Create vectorial forces map  Fx, Fy and DFx, DFy
[dim1,dim2]=size(W);


FX =1*ones(dim1,dim2);%(-arriba,+abajo)
%FX:,206:dim2)=-0.99;
FY =-2*ones(dim1,dim2);%(+izda,-dcha)
%N=ones(size(W));%
N=ones(size(W));%(W+2*alpha)*max(max(sqrt(FX.^2+FY.^2)));
```

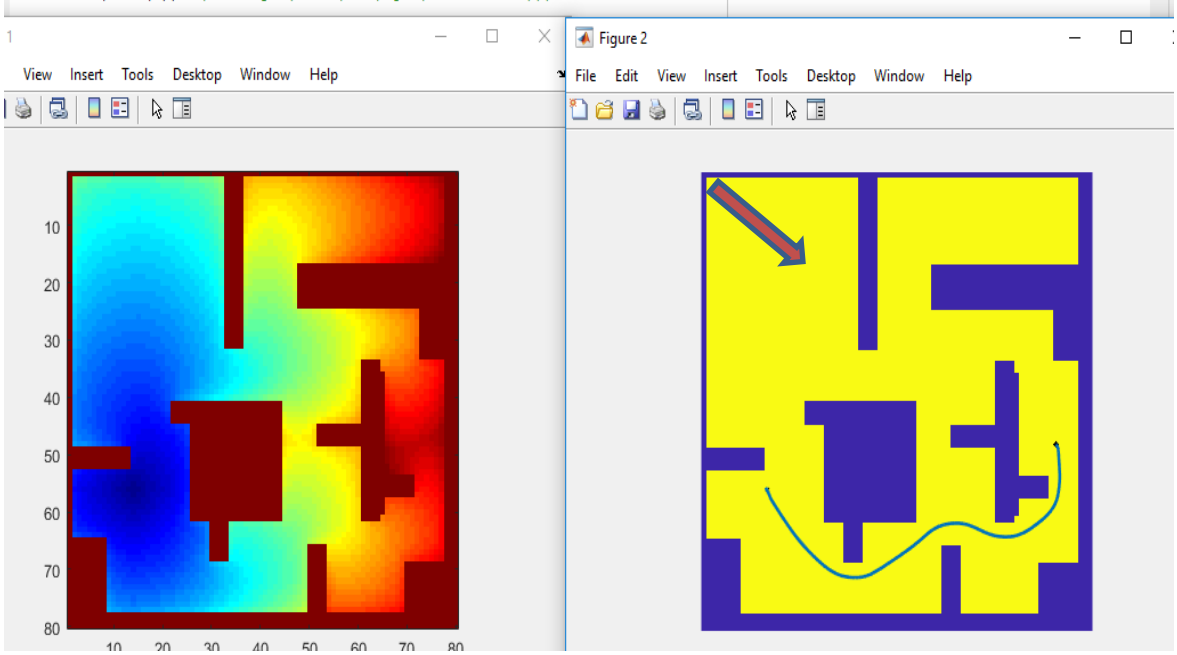


Figura 4.34: Room con corrientes (1,-2)

El significado físico de FX y FY no tiene misterio, es simplemente un campo vectorial que "empuja" a la trayectoria en un sentido u otro, igual que el viento puede acelerar, frenar o influir de cualquier modo en la trayectoria de un barco de vela. En este caso, al ser valores constantes, la representación de FX=1 sería:

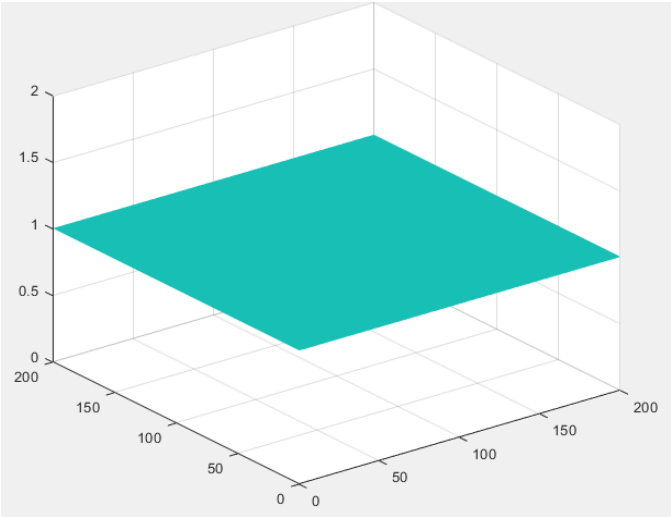


Figura 4.35: Representación de corriente FX=1

Como se ve, es la forma más simple de representación posible; sin embargo, este es el caso más sencillo y menos real ya que en la naturaleza la corriente no es constante, sino todo lo contrario. Y nuestro objetivo es aplicar Fast Marching a rutas marítimas con corrientes reales, por lo que más adelante daremos valores que existan y sean distintos en cada coordenada, dando FX de similares al siguiente:

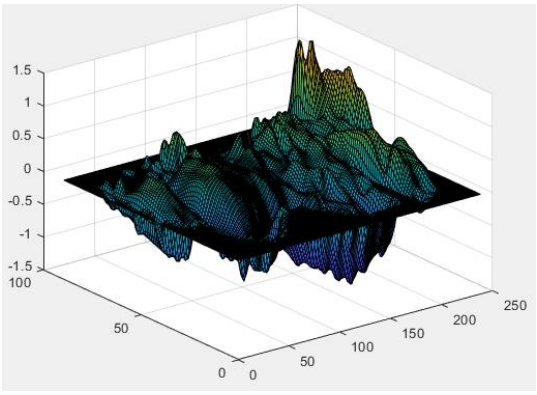


Figura 4.36: Representación de corriente FX real

Pero eso lo veremos más adelante. Ahora lo que nos interesa es mostrar como la variación de la corriente puede alterar completamente la trayectoria. Es importante remarcar que una corriente negativa en Y ha hecho que la trayectoria óptima sea por debajo en vez de por encima, es decir, que una corriente positiva en Y empuja las trayectorias hacia arriba, mientras que una negativa las arrastra hacia abajo. Como es de esperar, ocurre lo mismo con la coordenada X al variar FX, solo que afectando al movimiento horizontal en vez de al vertical.

4.3 TERCERA FASE

4.3.1 Búsqueda de datos externos.

Una vez controlado el funcionamiento de la *toolbox* de Fast Marching en Matlab, el siguiente paso es aplicarlo a un caso real de viaje marino con corrientes. Para ello son imprescindibles dos cosas:

- Un mapa real en blanco y negro que represente la superficie marina que será utilizada. El mapa debe contener las características de tamaño y forma vistas previamente para evitar errores como los comentados.
- Asociado a este mapa, y en sus mismas coordenadas, necesitamos un mapa de vectores que indiquen la corriente en cada punto; en cada coordenada dividiremos cada vector en su componente horizontal (FX) y vertical (FY) respectivamente.

Esta tarea no es nada sencilla, ya que por separado es muy fácil encontrar un mapa, y no es difícil encontrar una base de datos con los valores de las corrientes; pero lo realmente complicado es compaginar ambas cosas ya que la matriz corrientes debe estar en las mismas coordenadas que el mapa, pues evidentemente sino tendríamos una corriente que no se corresponde con la realidad del mar.

Con estos objetivos en mente, comencé a mirar en bases de datos como OSCAR (*Ocean Surface Current Analyses Real-time*) [20], un proyecto fundado por la NASA que permite ver en tiempo real (y con anterioridad) los mapas de corrientes marinas, vientos, temperatura, etc. Además de eso, también permite descargar los archivos masivos que contienen dicha información.

El descubrimiento de esta página en un principio parecía resolver las dificultades del presente proyecto, pero no fue así. Pese a que es una base de datos enorme y muy interesante, tiene dos problemas esenciales:

- Los datos aparecen sin un mapa al que haga referencia, luego es prácticamente imposible encontrar aquel que corresponde y lograr que tenga las mismas coordenadas y proporciones para que no se distorsione la información.
- El volumen de datos es demasiado grande, cuando hasta ahora en este proyecto la *toolbox* de FM solo ha funcionado bien con imágenes y matrices reducidas.

La parte positiva es que en esta búsqueda de una base de datos que nos fuera útil, dimos con una importante *toolbox* que me permitió realizar los dos primeros ejemplos con corriente real: *nctoolobox*; una herramienta que nos permite utilizar y acceder a archivos de datos tipo *nc* y que además viene con unos ejemplos que han sido esenciales para la continuación del trabajo.

Los dos ejemplos son los siguientes:

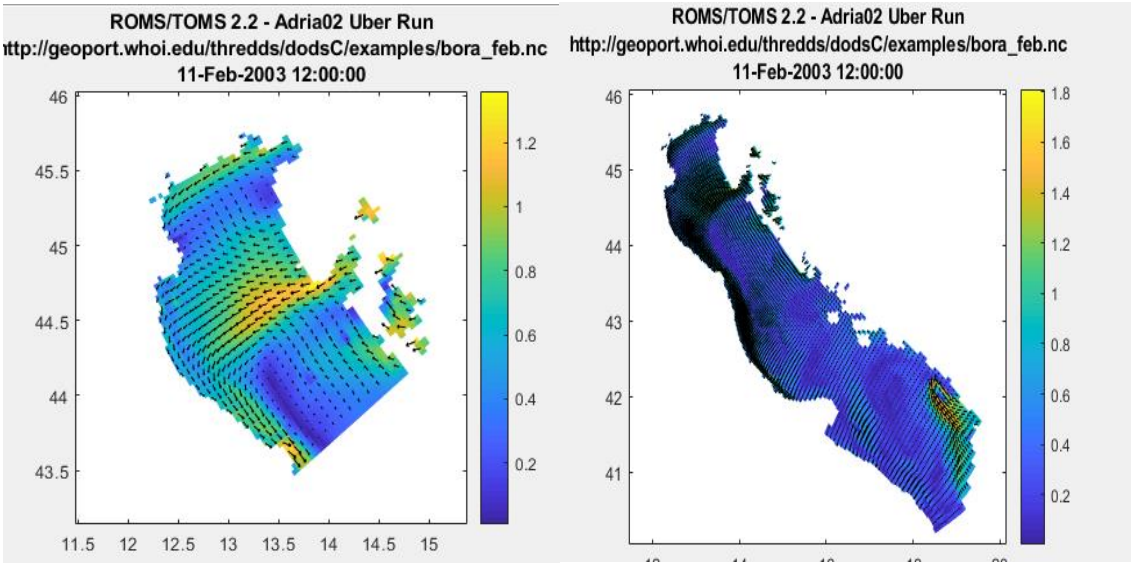


Figura 4.37: Imágenes de la función Geodemo_7 en nctoolbox

Estos mapas representan el mar Adriático:

Figura 4.38: Imagen satélite del mar Adriático *[21]*

En la *toolobox*, la figura de la derecha representa el mar en su totalidad, mientras que en la de la izquierda solo se representa la esquina superior izquierda. Lo importante es que con esta *toolbox* se accede al archivo *nc* real que contiene los datos de las corrientes y son representados en las imágenes, tanto por el color como por las flechas (vectores) según el sentido e intensidad de la corriente

Una vez que tenemos esto, podemos poner en práctica nuestro método con un ejemplo real, solo necesitamos dos cosas: el mapa y la matriz de corrientes.

Obtener las imágenes en blanco y negro es sencillo con las propias herramientas de Matlab:

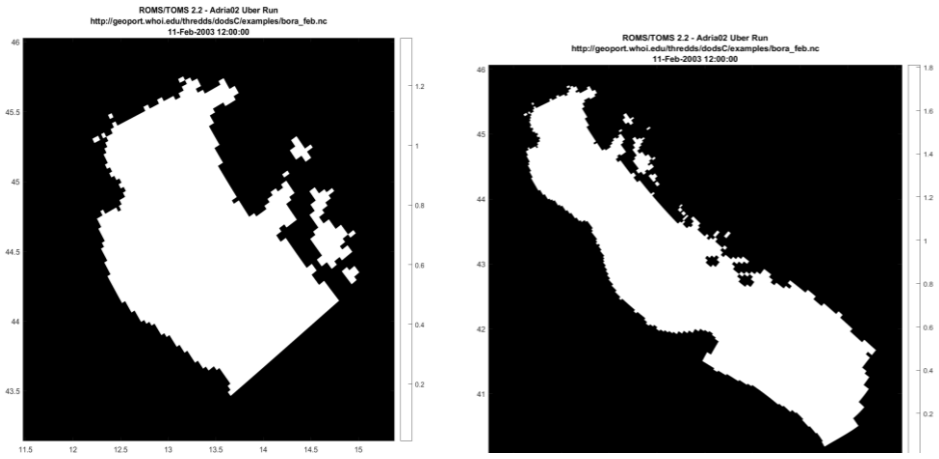


Figura 4.39: Imágenes del Mar Adriático en blanco y negro

Para obtener la matriz es más complicado ya que se representaba a través de números complejos y valores *NaN* (*Not a Number)* en las coordenadas de tierra, es decir, en la parte negra del mapa, lo que tiene sentido ya que en esos puntos no hay valor de corriente. Este era el aspecto original de un fragmento de la matriz:

19	20	21
NaN + NaNi	NaN + NaNi	NaN + NaNi
NaN + NaNi	NaN + NaNi	NaN + NaNi
NaN + NaNi	NaN + NaNi	NaN + NaNi
0.2772 - 0.3745i	0.3866 - 0.3047i	NaN + NaNi
0.2443 - 0.4834i	0.3169 - 0.4349i	0.3983 - 0.3828i
-0.0305 - 0.4000i	0.0964 - 0.4243i	0.2240 - 0.4297i
-0.2292 - 0.4743i	-0.0927 - 0.5197i	0.0668 - 0.5308i
-0.3582 - 0.5313i	-0.2784 - 0.6046i	-0.1388 - 0.6600i
-0.4551 - 0.5597i	-0.4158 - 0.6305i	-0.3327 - 0.6485i
-0.5540 - 0.5489i	-0.5343 - 0.6012i	-0.4679 - 0.5843i
-0.6418 - 0.5138i	-0.6300 - 0.5462i	-0.5483 - 0.5318i
-0.7302 - 0.4601i	-0.7231 - 0.4876i	-0.6143 - 0.4878i
-0.8096 - 0.4223i	-0.8061 - 0.4628i	-0.6853 - 0.4567i
-0.8718 - 0.4101i	-0.8747 - 0.4520i	-0.7484 - 0.4233i
-0.9347 - 0.4008i	-0.9417 - 0.4236i	-0.7991 - 0.3629i

Tabla 4.1- Matriz original Geodemo_7 en nctoolbox

Por lo tanto, lo primero que hubo que hacer fue transformar todos los *NaN* en ceros a través de la función *isnan*, la cual permite realizar una acción concreta con todos los *NaN* de la matriz*;* y después dividir los números complejos en su coordenada horizontal y vertical, que serán los valores de FX y FY respectivamente.

Para este último paso de nuevo Matlab tiene funciones que facilitan enormemente el trabajo, ya que a través de *vr=real(matriz)* y *vi=imag(matriz)*, conseguimos tener dos matrices, la vr y la vi, cada una de ellas teniendo solo la componente real o imaginaria (horizontal o vertical) respectivamente.

En las siguientes imágenes muestro como queda en mismo fragmento de matriz una vez he realizado las dos operaciones anteriormente descritas: los *NaN* son ahora ceros; y en vr tenemos los valores reales mientras que en vi los imaginarios. Además, es importante observar que los valores están entre 1 y -1, lo que es perfecto para FM ya que el código comienza a funcionar mal cuando los valores de corriente son muy superiores, en valor absoluto, a la unidad.

19	20	21
0	0	0
0	0	0
0	0	0
0.2772	0.3866	0
0.2443	0.3169	0.3983
-0.0305	0.0964	0.2240
-0.2292	-0.0927	0.0668
-0.3582	-0.2784	-0.1388
-0.4551	-0.4158	-0.3327
-0.5540	-0.5343	-0.4679
-0.6418	-0.6300	-0.5483
-0.7302	-0.7231	-0.6143
-0.8096	-0.8061	-0.6853
-0.8718	-0.8747	-0.7484
-0.9347	-0.9417	-0.7991

Tabla 4.2- Matriz Vr

19	20	21
0	0	0
0	0	0
0	0	0
-0.3745	-0.3047	0
-0.4834	-0.4349	-0.3828
-0.4000	-0.4243	-0.4297
-0.4743	-0.5197	-0.5308
-0.5313	-0.6046	-0.6600
-0.5597	-0.6305	-0.6485
-0.5489	-0.6012	-0.5843
-0.5138	-0.5462	-0.5318
-0.4601	-0.4876	-0.4878
-0.4223	-0.4628	-0.4567
-0.4101	-0.4520	-0.4233
-0.4008	-0.4236	-0.3629

Tabla 4.3- Matriz Vi

Una vez realizado esto, aplicamos la funcion *imresize* para hacer que cada trío mapa-FX-FY sea cuatro veces más grande (ya que la imagen de partida era muy pequeña) y esté en el intervalo entre 100x100 y 200x200 píxeles, que es el idóneo. Una vez realizado esto, los resultados se muestran a continuación.

Para la imagen de menor tamaño, es decir, la que solo muestra el tramo final del mar Adriático:

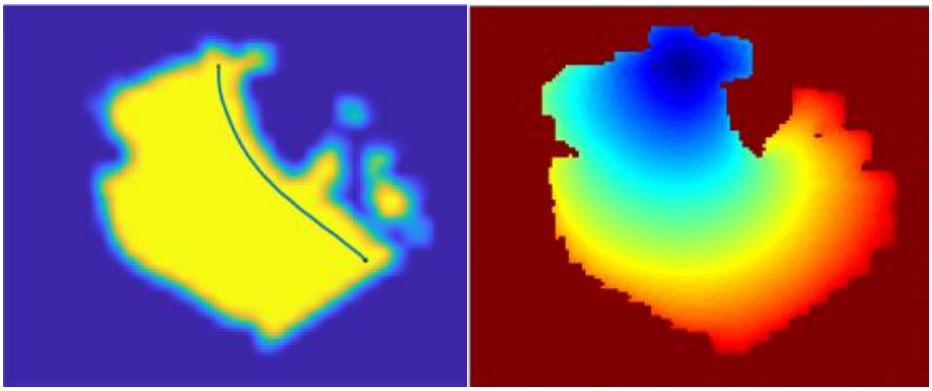


Figura 4.40: Trayecto y tiempo de llegada en már Adríatico 1

La calidad obtenida no es la óptima, siendo visible un cierto halo azulado que rodea al perímetro del mar, lo que muestra ciertas complicaciones en el cálculo del FM; sin embargo, el trayecto es trazado correctamente y el tiempo de llegada da un resultado coherente e intuitivo.

Lo más interesante llegados a este punto es el valor de las corrientes reales, y ver que se corresponden con la geografía:

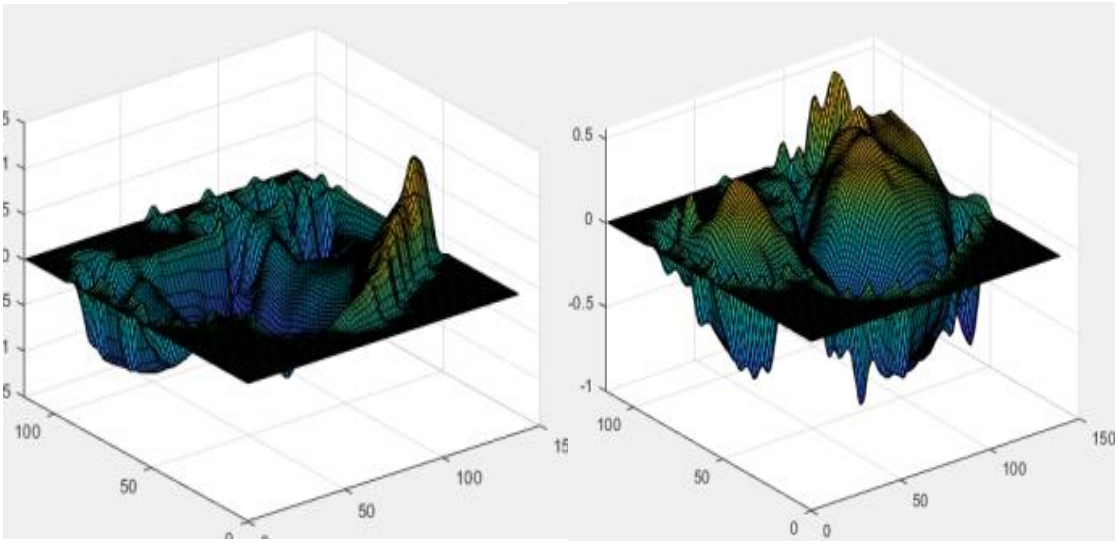


Figura 4.41: FX y FY en mar Adriático 1

Para el mapa del mar Adriático en toda su extensión obtenemos lo siguiente:

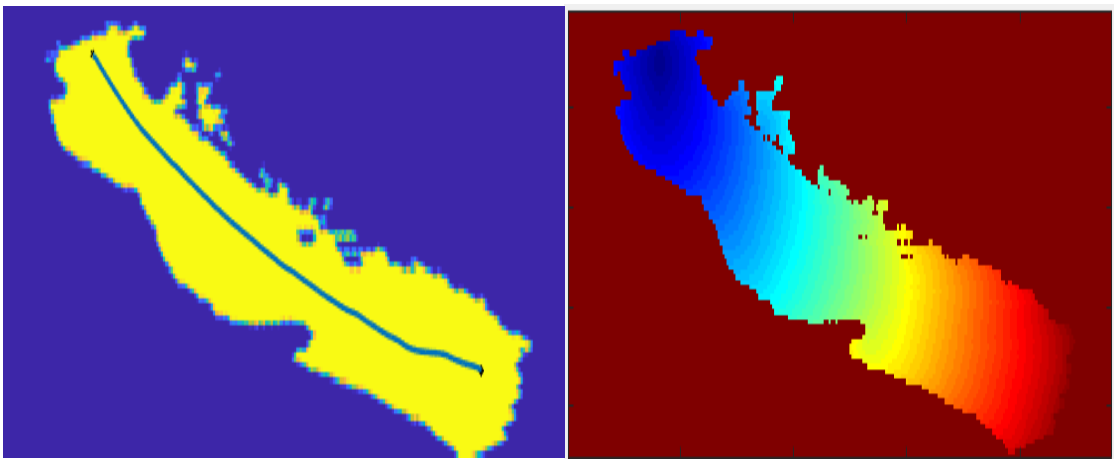


Figura 4.42: Trayecto y tiempo de llegada en már Adríatico 2

En cuanto a las corrientes, en este segundo caso tienen la siguiente apariencia:

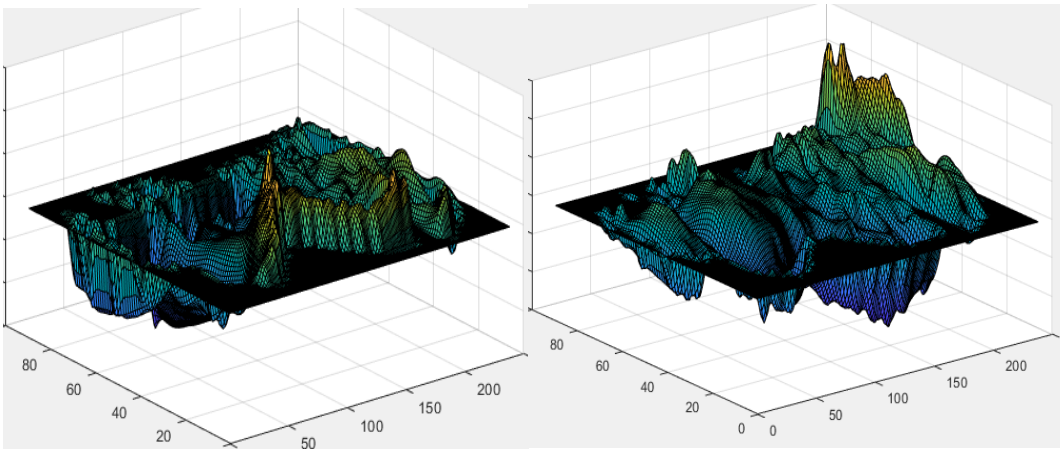


Figura 4.43: FX y FY mar Adriático 2

Por último, representando el tiempo de llegada en el eje vertical, obtenemos la siguiente figura:

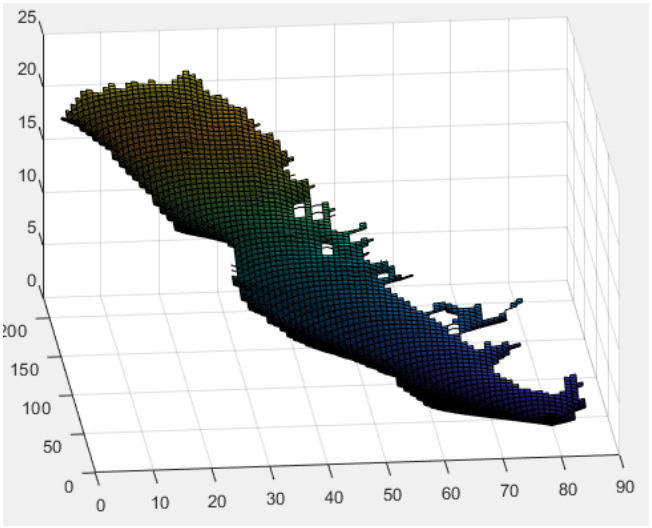


Figura 4.44: Tiempo de llegada en mar Adriático 2

4.3.2 Flypath

En la fase 1 comprobamos el funcionamiento del FM con ejemplos sencillos que no tenían correspondencia con la realidad, simplemente para poner a prueba el método. En la fase 2 estudiamos cómo funcionan las corrientes, y en el apartado anterior comprobamos el correcto funcionamiento del FM cuando se aplica a corrientes y mares reale. Ahora vamos a dar el último paso: utilizar Fast Marching en mapas reales y representar la trayectoría de una manera más visual.

Para ello, de nuevo es necesario apoyarnos en fuentes externas, y en este caso es esencial la *toolbox* de *flypath* , un paquete de matlab de libre uso desarrollado por el polaco Witold Bużantowicz.

Esta *toolbox* es realmente potente e interesante, ya que permite realizar visualizaciones 2D y 3D de barcos, aviones, lanzamientos de misiles etc. [22]Además, permite trazar las trayectorias e incluso simular colisiones, incluyendo un gran número de figuras para tener distintos modelos de cada tipo en cuestión.

En nuestro caso, solo vamos a utilizar *flypath* para trazar el trayecto en el mapa ayudándonos de las funciones y elementos de dicha herramienta, pero tiene mucho más potencial y desde luego es muy interesante para casi cualquier aplicación de *Matlab* ya nos permite simular un gran número de trayectos, desde el vuelo de un avión hasta el movimiento de un robot a la hora de encontrarse obstáculos en su camino.

En este proyecto, haremos uso de uno de los modelos más basicos de barco que tiene la *toolbox*:

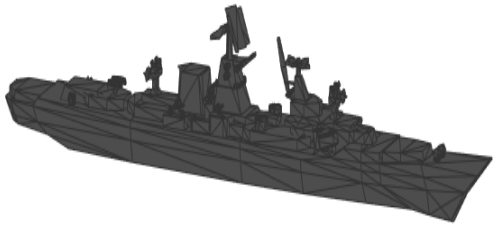


Figura 4.45: Barco flypath

Pero también incluye otros modelos muy elegantes, como el siguiente caza, que puede ser perfectamente utlizado para las aplicaciones de FM en el aire:

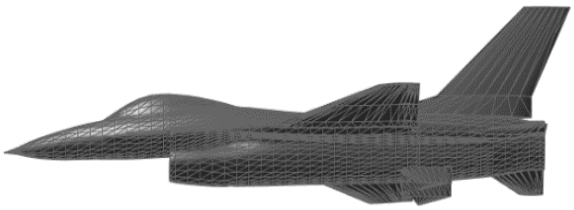


Figura 4.46: Caza flypath

Se puede comprobar la potencia de la *toolbox* en el siguiente ejemplo, donde observamos distintos tipos de barcos, de cazas y hasta un helicóptero realizando distintas maniobras:

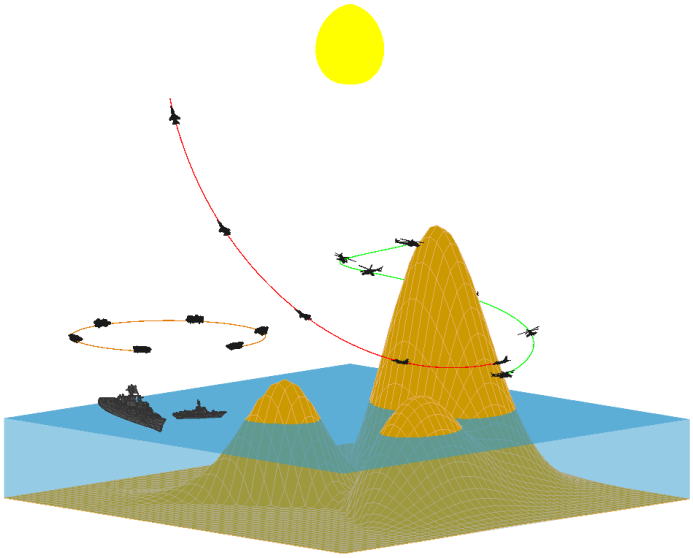


Figura 4.47: Escena naval flypath

Por último, añadir que estas simulaciones se pueden hacer dinámicas de forma que los elementos vayan moviéndose en el tiempo, pero por el contenido y forma de este trabajo me limitaré a una simulación estática, teniendo en cuenta que es una herramienta muy útil de cara a futuras mejoras y ampliaciones.

4.3.3 Atazar

Teniendo como base todo lo visto a lo largo del proyecto, es el momento de ponerlo a prueba en un mapa real y simular el trayecto sobre el mismo. Para ello se tomará el embalse de El Atazar en Madrid como imagen original; como siempre, también será necesario su equivalente en blanco y negro:

Figura 4.48: Imagen de El Atazar

Eliguiendo dos puntos suficientemente separados, este es el resultado:

Figura 4.49: Trayectoria 1 en Atazar

El trayecto se ha trazado correctamente, pero nunca es mal momento para comprobar el funcionamiento a través de otras vías. Por ejemplo analizando el primer potencial se puede observar un resultado coherente ya que el coste de llegada es menor cuanto mas alejado esta de la costa, es decir, que son preferibles las coordenadas que están más dentro del embalse tanto por seguridad como por suavidad de la trayectoria.

En la figura de la izquierda vemos el potencial en su forma clásica, y en la derecha se incluye un tercer eje para hacer más visual el hecho de que los puntos altos (a modo de pico de montaña) son los óptimos, mientras que se va penalizando el resto de coordenadas según disminuyen la altura. Es decir, el color oscuro representa menos altura, mientras que el blanco representaría una altura infinita.

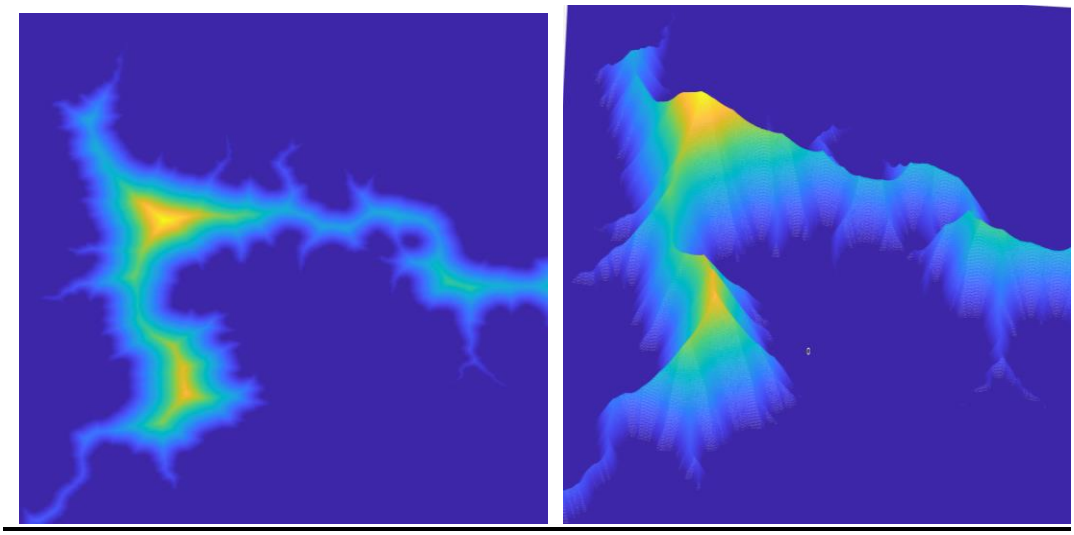


Figura 4.50: Primer potencial en Atazar 1

La otra comprobación se realizará a través del tiempo de llegada, teniendo el resultado esperado: alrededor de la zona de salida el resultado es azul (poco tiempo) y se va volviendo amarillo según se aleja del punto de origen.

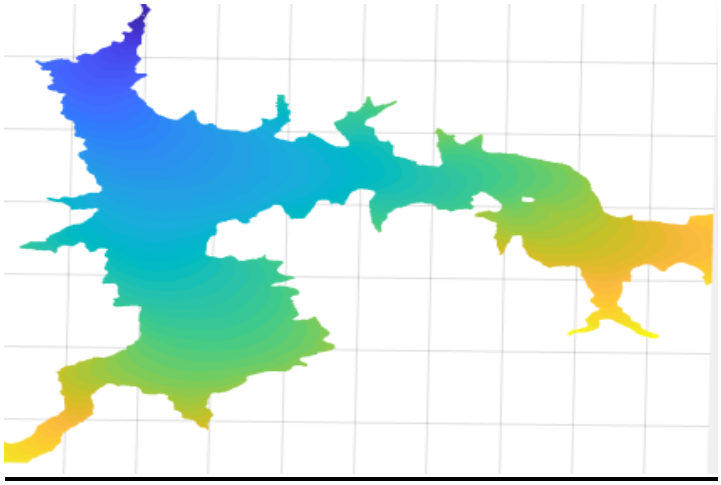


Figura 4.51: Tiempo de llegada en Atazar 1

Representando el tiempo como eje vertical da un resultado elegante ya que las dos ramas del embalse se abren según la onda se expande a lo largo de ellas, dando lugar a una figura peculiar:

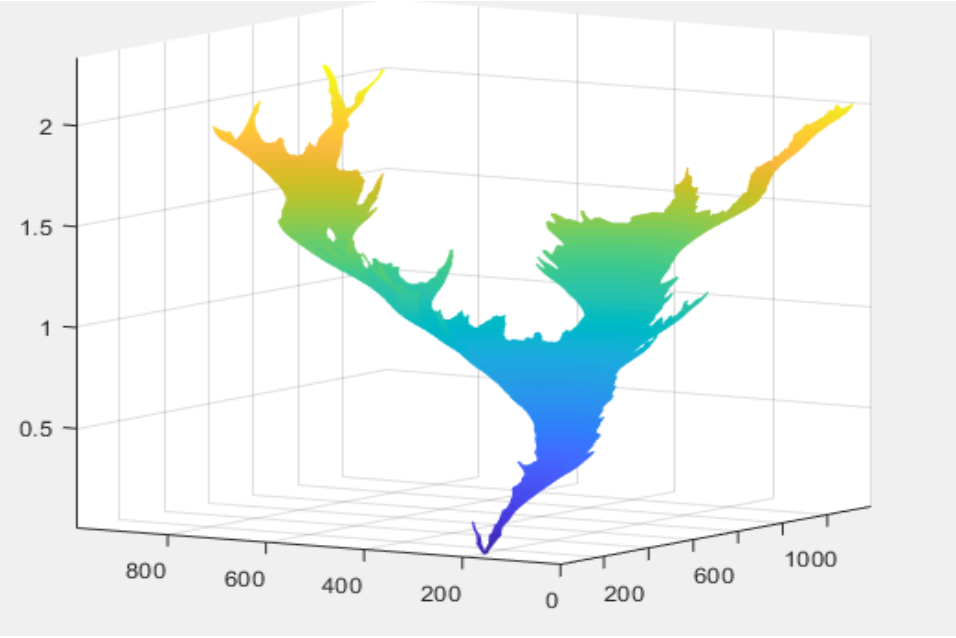


Figura 4.52: Tiempo de llegada vertical en Atazar 1

Se comprueba con otro trayecto y el resultado sigue siendo satisfactorio:

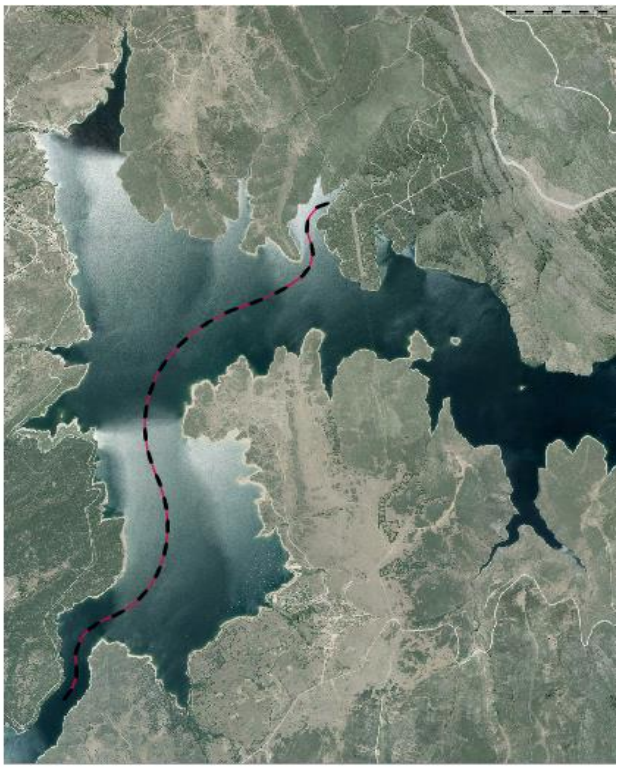


Figura 4.53: Trayectoria en Atazar 2

El tiempo de llegada vuelve a ser intuitivo y lo que se esperaba de él:

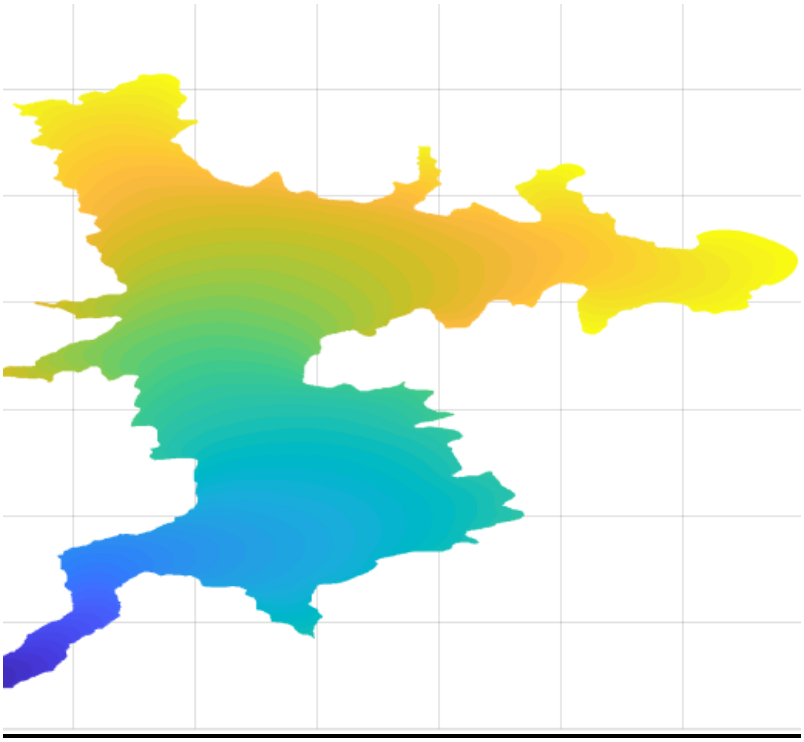


Figura 4.54: Tiempo de llegada en Atazar 2

Por último, podemos aprovechar estos avances para mostrar el trayecto sobre el mar Adriático con el barco sobre la imagen de satélite:

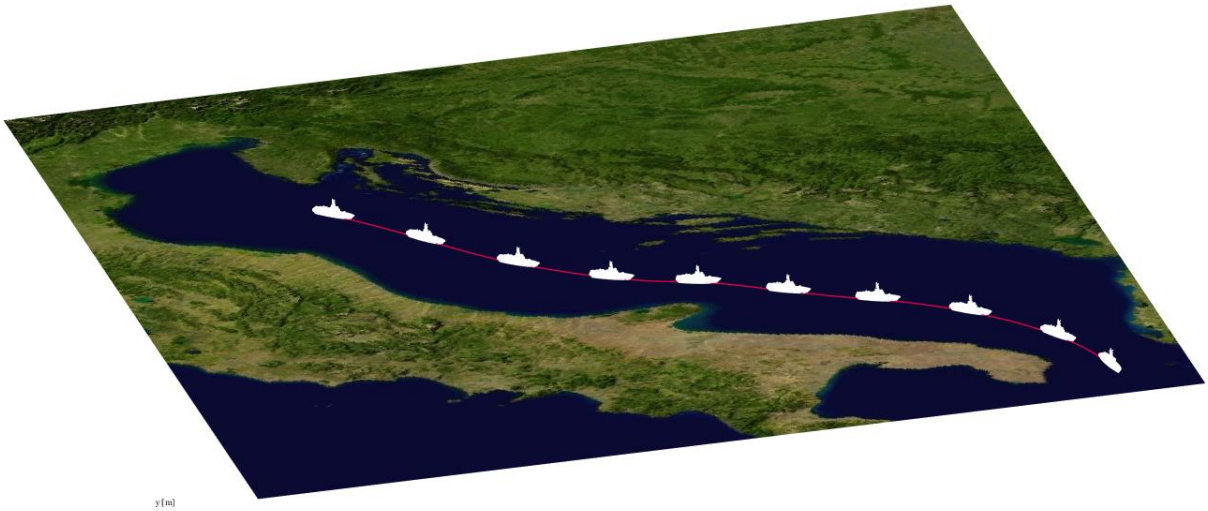


Figura 4.55: Trayecto sobre mar Adriático con imagen satélite

5.Trabajo futuro

Uno de mis objetivos a lo largo de este trabajo ha sido mostrar la potencia del FM y de la herramienta de Matlab a través de varias *toolbox* con las cuales realmente se puede llegar a hacer simulaciones y análisis muy variados. En cuanto a trabajos futuros, veo principalmente cuatro líneas de trabajo:

- Por un lado, este método y prácticamente la totalidad del trabajo aquí desarrollado es perfectamente extrapolable a otros campos como trayectorías de robots, viajes submarinos, vuelos de aviones etc. Incluso profesores de la UC3M han aplicado estas técnicas para para simular el desplazamiento de robots sobre la superficie de Marte [23]. Por lo tanto, se debe seguir investigando en paralelo las aplicaciones del FM en una gran cantidad de líneas de trabajo diferentes.

- Por otro lado, con este proyecto se ha mostrado que el método funciona correctamente con datos de corriente real y sobre mapas también existentes. Sin embargo, no se pudo explorar más allá de los ejemplos aquí presentados ya que, pese a varias semanas de búsqueda, es muy complicado encontrar una matriz de vectores con los datos de corriente de un lugar y, junto a esta, el mapa del mismo lugar. Por separado no es complicado, pero encontrarlo junto sí lo es. Y es necesario tener un mapa de vectores que esté en las mismas coordenadas y en la misma escala que otro mapa. Por lo tanto, esta línea de trabajo será muy interesante siempre que sea posible encontrar tales datos en el formato necesario para que pueda ser aplicado al FM.

- Se puede profundizar en los niveles de saturación o factores de seguridad. En este trabajo no me he centrado en este punto, realmente lo único que significa a la hora de la verdad es que el primer potencial será más prohibitivo, es decir, que las zonas grisáceas entre las barreras (negro) y las zonas libres (blanco) serán más amplias, provocando que la trayectoria tenga que pasar más alejado de dichos límites.

- Por último, en futuras mejoras debería tenerse en cuenta la curvatura de la tierra ya que en un viaje corto apenas se notará, pero según se aumenta la distancia recorrida el error cometido es mayor y ya no valdría con representar los mapas como superficies planas sino que habría que incorporar esta nueva variable.

6. Presupuesto

El trabajo ha sido desarrollado completamente a través de Matlab, por lo que el presupuesto en este caso ha sido nulo ya que he podido utilizarlo de forma gratuita al tener una licencia académica gracias a la UC3M. Sino, tal como figura en la página web de Matlab [24], una licencia estándar anual me hubiese costado 800 €, o 2000 € una vitalicia; y tal sería el presupuesto para el trabajo en caso de no contar con dicha licencia.

Aparte del mencionado software, lo único utilizado han sido los artículos de investigación (principalmente de investigadores de la UC3M) y otras fuentes de datos citadas en las próximas páginas, y varias *toolbox* de libre acceso absolutamente imprescindibles para lograr los objetivos.

7. Conclusiones

Ha sido un proceso largo y complicado desde que comencé a trabajar en él en noviembre del año pasado leyendo los artículos de divulgación de mi tutor y otros profesores. La base matemática y física que hay detrás es muy compleja como para entenderla completamente en estos niveles, pero la aplicación experimental es más intuitiva y sencilla por lo que he podido realizar el trabajo sin grandes problemas conceptuales, sino que más bien estos han sido problemas técnicos.

A lo largo de estos meses, me he encontrado con dos dificultades principales:

- Por un lado, nunca había utilizado Matlab, por lo que he aprendido a trabajar con el sofware desde el inicio con funciones y herramientas complejas enfrentándome a una curva de dificultad que en los primeros meses era muy elevada pues desconocía por completo el funcionamiento del programa.

  Además, diversas funciones y aplicaciones de ciertas *toolbox* han estado provocando errores de tipos muy distintos y en algunos casos tuvieron que pasar semanas hasta que logré solucionarlo, ya fuera gracias a la ayuda inestimable de mi tutor, de otros compañeros más curtidos en el programa, o de páginas web. En ciertos códigos me encontraba errores casi a cada línea de comando y, del mismo modo que era frustrante no lograr ni el funcionamiento más básico, también ha sido gratificante el ir alcanzando los objetivos y obteniendo los resultados esperados. Esta etapa correspondería a la fase 1 del proyecto, aunque realmente la gran mayoría de ejemplos y problemas no los he incluido en el trabajo al considerar que no aportaban nada nuevo o relevante.

- Una vez controlado el funcionamiento del FM en Matlab, el mayor desafío fue aplicarlo a mapas reales con las corrientes correspondientes. Como ya mencioné, encontrar una matriz númerica de corriente correspondiente a un mapa concreto ha sido una misión muy complicada pese a estar meses dedicado a ello. Hay bases de datos que muestran las corrientes en tiempo real (como el proyecto *OSCAR* de la *NASA)*, incluso archivos nc que contienen esas magnitudes, pero no era aplicable en nuestro código ni contenía la imagen a la que hacía referencia.

  Esto provocaba una sensación de cierta frustración, pero finalmente he conseguido mostrar el correcto funcionamiento del FM con varios ejemplos de corriente real, por lo que añadir más casos a modo de ejemplo realmente no aportaría nada nuevo al proyecto.

En general, ha sido un trabajo en el que se ha puesto a prueba la resistencia y la constancia, ya que en muchas etapas parecía que no se podía avanzar por diversas incidencias, y solo mi tutor Santiago sabe bien el número de dificultades que he ido encontrando.

En todas las facetas de la vida, tanto en lo académico, como en lo personal y profesional, es imprescindible tener una suficiente fortaleza mental como para persistir en lograr los objetivos propuestos y no dejarse vencer por la frustración o por verse superado ante dificultades que en principio uno no es capaz de resolver. Por esto, el haber trabajado por primera vez en un proyecto tan largo, me ha permitido vivir una experiencia interesante de cara al futuro, ya que volveré a enfrentarme a trabajos similares y seguramente más largos y complejos por lo que toda preparación es necesaria y bienvenida.

Como mencioné en el apartado del presupuesto, si no fuera por la licencia gratuita de matlab me hubiese sido imposible aprender dicho programa y utilizarlo, por lo que no hubiese podido realizar el trabajo. Por esta y por muchas otras cosas defenderé siempre la educación pública en general y a la UC3M en concreto, ya que en estos cuatro años de carrera he podido aprender de grandes profesores, emplear potentes servicios y vivir valiosas experiencias.

Como últimas palabras, he de decir que estoy satisfecho con el proyecto y que se ha logrado el objetivo inicial: aplicar el método del Fast Marching en trayectos marinos con datos de corrientes reales. Por el camino, he podido explicar y mostrar el funcionamiento del FM y de las corrientes, además de añadir trayectos con barcos sobre imágenes reales. Por todo esto, considero que se han logrado buenos resultados y se ha dejado abierta la puerta a futuros desarrollos.

Apéndice: fragmentos de diversas funciones

A continuación muestro una versión simplificada de una función utilizada innumerables veces para trazar una trayectoria básica con FM:

Código FM vectorial

```
clear
map_name = 'room_b.png';
[Wo, cm] = imread(map_name);
Wo=flip_vertical(Wo(:,:,1));
Wo = Wo';

Wo3=bwdist(~Wo);
Wo3 = double(Wo3) ;
Wo3=rescale( smoothn(Wo3));
figure(1)
% para representacion
W= rescale( double(Wo3) );

start_points=[34;255];
      end_points=[634;263];
options.nb_iter_max = Inf;
options.end_points = end_points;
alpha=0.1;

% New vectorial fields thing
% Create vectorial forces map  Fx, Fy and DFx, DFy
[dim1,dim2]=size(W);
FX =1*ones(dim1,dim2);

FY =1*ones(dim1,dim2);
%N=ones(size(W));%
N=(W+2*alpha)*max(max(sqrt(FX.^2+FY.^2)));
disp('Performing front propagation.');
[D,S] = fast_marching_vectorial_2d(W, start_points, options, FX,FY, N);
figure
imagesc(flip_vertical(W'));
axis equal
axis tight

figure
imagesc(flip_vertical(D'));
colormap jet
axis equal
axis tight
figure
disp('Extracting path.');
path = extract_path_2d(D,end_points, options);
plot_path_2d(Wo,S,path,start_points,end_points);
```

Por último, muestro el código utilizado para trazar la trayectoria sobre imágenes reales:

Código trayectoria real

```
%   Path planning utilizando fast marching y la transformada de distancia
%
n = 200;
%Psi0 = 180*pi/180;  % rad (ang timon barca)

% Robot radius, cell size, radius in cells, diameter in cells
global r_radius;
r_radius=12*5; %centimeters metros para barco
global cell_size;
cell_size=12.1;  % en centimetros valor de defecto
global d_r_shape
d_r_shape=ceil(2*r_radius/cell_size);
global r_r_shape
r_r_shape=ceil(r_radius/cell_size);

num_sensores=721;%181; %361 medidas;
incr_th0=0.5*pi/180;   %0.5; % incremento entre sensores en grados
cell_size=12.1*5;  % en centimetros
s_range=2000/cell_size;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Parametros del modelo
rate_max = 10*pi/180;   % max rudder rate  rad/seg
delta_max = 30*pi/180;  % max rudder angle rad
tauv = 5;               % cts de tiempo velocidad barco

% Parametros viento
Vw = 0.0;           % wind speed m/s
psiw = 45*pi/180;     % wind heading  rad

% Parametros desalineamiento
desx = 0.0; %0.3;          % desalineamiento x  m
desy = -0.0; %-0.1;         % desalineamiento y  m
desang = 0.0*pi/180; %1*pi/180;    % desalineamiento angular rad

% Parametros del controlador
Kp = 1.5;
Kd = 1.3529;
Km =0.6797;
Kfm = 5;
wfm = 5;
Kr = 0.3036;  % FT del barco 0.3036/(s+0.7) se usa en el control
pr = 0.7;

% control discreto
h = 0.2;
Cd = c2d(tf([Kfm,0],[1,wfm]),h);
[numf,denf]=tfdata(Cd,'v');
  CONTROL = 1;       % Control discreto



% map_name = 'atazar.bmp';
% [Wo, cm] = imread(map_name);
%
% bw=imread('uc3m_p3.bmp');
```

```
%imcolor=imread('A2_VertBWs.png');%B/N
%imcolor2=imread('A2_Vertical1s.png');% color
imcolor=imread('IM2.png');%B/N
imcolor2=imread('mar.png');% color
imcolor=flip_vertical(imcolor);
imcolor2=flip_vertical(imcolor2(:,:,2));
W2=imcolor2';

bw = im2bw(imcolor,0.4);
Wo = bw;
%imshow(imcolor), figure, imshow(bw);

SE=strel('disk',1);
bw3 = imdilate(~bw,SE);
bw4=~bw3;
% figure;imshow(bw4);
% pause

%tic;
%W=bwdist(~Wo');
W=bwdist(~bw4');
%toc;
W = rescale( double(W) );
W(isnan(W))=0;
W(isinf(W))=0;
%W=smooth3(W);
%W = rescale( double(W) );
%W = W;

% para representacion
W1= rescale( double(Wo') );

options.nb_iter_max = Inf;
options.end_points = end_points;
options.Tmax = sum(size(W));
 W3=real(W.^0.3);%saturation
[D,S] = perform_fast_marching_2d(W3, start_points, options);
path = extract_path_2d(D,end_points, options);
Psi0= atan2(path(length(path)-100,2)-path(length(path),2),path(length(path)-100,1)-
path(length(path),1));
Xk_k(3)= Psi0*180/pi ; %grados
sz=size(cdata);
path2=zeros(length(path),6);
path2(:,1)=path(:,1);
path2(:,2)=sz(1)-path(:,2)+1;

path2(:,3:6)=0;
[path2(:,6),path2(:,4),path2(:,5)] = xyz2rpy(path2(:,1),path2(:,2),path2(:,3));
path2(:,4)=path2(:,4)*pi/180;path2(:,5)=pi-path2(:,5)*pi/180;path2(:,6)=path2(:,6)*pi/180;

new_object('tarantula.mat',path2,'model','tarantula_corvette2.mat','scale',.4,...
'edge',[1 1 1],'face',[1 1 1],'alpha',1,...
'path','on','pathcolor',[.89 .0 .27],'pathwidth',1);

 flypath('tarantula.mat','animate','off','step',400,...
'axis','on','axiscolor',[0 0 0],'color',[1 1 1],...
'font','Georgia','fontsize',6,...
'view',[0 90],'window',[1800 900],...
'xlim',[1 1000],'ylim',[1 1325],'zlim',[-10 10]);
```

BIBLIOGRAFÍA Y REFERENCIAS

[1] Gómez, Javier & Lumbier, Alejandro & Garrido, Santiago & Moreno, Luis. (2013). Planning Robot Formations with Fast Marching Square including Uncertainty Conditions. Robotics and Autonomous Systems. 61. 137-152. 10.1016/j.robot.2012.10.009.

[2] Régis Monneau. Introduction to the Fast Marching Method. 2010.

[3] Jose Pardeiro Blanco (2015) Algoritmos de planificación de trayectorias basados en Fast Marching Square (tesis de máster), universidad Carlos III de Madrid, Madrid

[4] IConnell, D., & Manh La, H. (2018). Extended rapidly exploring random tree–based dynamic path planning and replanning for mobile robots. International Journal of Advanced Robotic Systems.

[5] MathWorks. "Toolbox Fast Marching". https://es.mathworks.com/matlabcentral/fileexchange/6110-toolbox-fast-marching (acceso: 27 de octubre de 2018).

[6] Witold Bużantowicz. "Flypath 3d". wbint. http://www.wbint.pl/flypath3d/index.php (acceso: 10 de diciembre de 2018).

[7] B. Schlining, R. Signell, A. Crosby, nctoolbox (2009), Github repository, https://github.com/nctoolbox/nctoolbox (acceso: 20 de enero de 2018).

[8] Valero-Gomez, Alberto & Gómez, Javier & Garrido, Santiago & Moreno, Luis. (2013). Fast Marching Methods in Path Planning. IEEE Robotics &amp amp amp Automation Magazine. 20. 111 - 120.

[9] R Nave. "Maxwell's Equations". Hyperphysics. http://hyperphysics.phy-astr.gsu.edu/hbase/electric/maxeq.html (acceso: 27 de diciembre de 2018).

[10] Graham Turnbull. "Maxwell's Equations". Ethw. https://ethw.org/Maxwell%27s_Equations. (acceso: 27 de diciembre de 2018).

[11] José Antonio Martín. "Ecuaciones de Maxwell" https://joseantoniomartin.wordpress.com/2016/02/10/ecuaciones-de-maxwell/ (acceso: 27 de diciembre de 2018).

[12] B.Surendranath. " Fermat`s principle". Surendranath. http://www.surendranath.org/GPA/Optics/Fermat/Fermat.html (acceso: 29 de diciembre de 2018).

[13] Garrido, Santiago & Moreno, Luis & Gómez, Javier & Lima, Pedro. (2013). General Path Planning Methodology for Leader-Follower Robot Formations. International Journal of Advanced Robotic Systems. 10. 10.5772/53999.

[14] Garrido, Santiago & Gómez, Javier & Alvarez, David & Moreno, Luis. (2014). Fast Marching with Restrictions.

[15] Valero-Gomez, Alberto & Gómez, Javier & Garrido, Santiago & Moreno, Luis. (2013). The Path to Efficiency: Fast Marching Method for Safer, More Efficient Mobile Robot Trajectories. Robotics & Automation Magazine, IEEE. 20. 111-120. 10.1109/MRA.2013.2248309.

[16] E.D. Solomentsev "Lyapunov surfaces and curves". Encyclopedia of math. https://www.encyclopediaofmath.org/index.php/Lyapunov_surfaces_and_curves. (acceso: 10 de enero de 2019).

[17]Luis moreno y Santiago Garrido. "Fast Marching Method: Application of the Eikonal equation in path planning problems". Canal Uned. https://canal.uned.es/uploads/material/Video/49784/Presentaci__n_Luis_Moreno.pdf (acceso: 2 de marzo de 2019).

[18] Gómez, Javier & Lumbier, Alejandro & Garrido, Santiago & Moreno, Luis. (2013). Planning Robot Formations with Fast Marching Square including Uncertainty Conditions. Robotics and Autonomous Systems. 61. 137-152. 10.1016/j.robot.2012.10.009.

[19] ElenVD. "black silhouette country borders map of Serbia on white background of illustration".shutterstock. https://www.shutterstock.com/image-illustration/black-silhouette-country-borders-map-serbia-1104009362. (acceso: 12 de febrero de 2019).

[20] NASA. "OSCAR". ESR (Earth & Space Research). https://www.esr.org/research/oscar/ (acceso: 20 de febrero de 2019).

[21] NASA. "File:Adriatic Sea.jpg". Wikimedia commons. https://commons.wikimedia.org/wiki/File:Adriatic_Sea.jpg. (acceso: 29 de febrero de 2019).

[22] Bużantowicz, Witold. (2016). Matlab Script for 3D Visualization of Missile and Air Target Trajectories. 5. 419-422.

[23]Garrido, Santiago & Moreno, Luis & Martín, Fernando & Alvarez, David. (2017). Fast Marching subjected to a Vector Field - path planning method for Mars rovers. Expert Systems with Applications. 78. 334-346. 10.1016/j.eswa.2017.02.019.

[24] The MathWorks, Inc. "Pricing and Licensing". MathWorks. https://es.mathworks.com/pricing-licensing.html (acceso: 10 de abril de 2019).